AdvaSoft® 2.0 for Windows NT

Advant History

Version 1.0

User's Guide

3BUF 000 297 R0001

Use of **DANGER**, **WARNING**, **CAUTION**, and **NOTE**

This publication includes, **DANGER**, **WARNING**, **CAUTION**, and **NOTE** information where appropriate to point out safety related or other important information.

DANGER	Hazards which could result in severe personal injury or death
WARNING	Hazards which could result in personal injury
CAUTION	Hazards which could result in equipment or property damage
NOTE	Alerts user to pertinent facts and conditions

Although **DANGER** and **WARNING** hazards are related to personal injury, and **CAUTION** hazards are associated with equipment or property damage, it should be understood that operation of damaged equipment could, under certain operational conditions, result in degraded process performance leading to personal injury or death. Therefore, comply fully with all **DANGER**, **WARNING**, and **CAUTION** notices.

TRADEMARKS

Advant, AdvaInform and AdvaBuild are registered trademarks of ABB Automation AB.

MOD 300, Batch 300, and Taylor are Trademarks of ABB Automation AB.

ORACLE, SQL*Plus are registered trademarks of Oracle Corporation.

NOTICE

The information in this document is subject to change without notice and should not be construed as a commitment by ABB Automation AB. ABB Automation AB. assumes no responsibility for any errors that may appear in this document.

In no event shall ABB Automation AB. be liable for direct, indirect, special, incidental or consequential damages of any nature or kind arising from the use of this document, nor shall ABB Automation AB. be liable for incidental or consequential damages arising from use of any software or hardware described in this document.

This document and parts thereof must not be reproduced or copied without ABB Automation AB.'s written permission, and the contents thereof must not be imparted to a third party nor be used for any unauthorized purpose.

The software described in this document is furnished under a license and may be used, copied, or disclosed only in accordance with the terms of such license.

Copyright © 2000 ABB Automation AB. All rights reserved.

TABLE OF CONTENTS

Chapter 1 - Introduction

1.1 About This Book 1
    1.1.1 How to Use this Book 1
1.2 Product Overview 2
    1.2.1 What You Can Do with Advant History 3
        1.2.1.1 History Database Configuration 3
        1.2.1.2 Data Collection 4
        1.2.1.3 Data Storage 5
        1.2.1.4 Data Retrieval 5
    1.2.2 What's New in this Release 6
    1.2.3 Release History 6
1.3 Prerequisites & Requirements 6
1.4 Conventions 7
1.5 Related Documentation 8
1.6 Terminology 10

Chapter 2 - Installation

2.1 Product Verification 13

Chapter 3 - Configuration

3.1 Getting Started 15
    3.1.1 The Composite Log Structure 17
    3.1.2 User Interface 18
        3.1.2.1 Available Composite Logs List 18
        3.1.2.2 Tree View 20
        3.1.2.3 Log Attribute Tabs 23
        3.1.2.4 Configuration Status Indication 23
        3.1.2.5 Main Menu Bar 25
    3.1.3 Application Start-up 29
3.2 Configuring Composite Logs via the History Builder 29
    3.2.1 Getting the Full Log List 30

CONTENTS (continued)

3.2.2 Customizing the Tree View 30
  3.2.2.1 Select Which Attributes to Show for Primary/Secondary Logs 31
  3.2.2.2 Show or Hide the Composite Log Attributes 32
  3.2.2.3 Expand the Tree View 32
3.2.3 Adding a Composite Log 32
3.2.4 Adding Numeric Logs to the Composite Log 35
  3.2.4.1 Simple Numeric Logs for Synchronous Process Data 35
  3.2.4.2 Secondary Logs for Different Resolution/Time Span Views 44
  3.2.4.3 Primary and Secondary Logs with Calculations 51
3.2.5 Deleting Logs 54
3.2.6 Adjusting Numeric Log Attributes 54
  3.2.6.1 Configuring Attributes from the Tree View 55
3.2.7 Starting/Stopping Logs 56
3.2.8 Log Attributes 56
  3.2.8.1 Numeric Log Attributes 57
  3.2.8.2 Calculation Attributes 59
  3.2.8.3 Deadband Attributes 66
  3.2.8.4 Alarm Attributes 71
  3.2.8.5 Data Collection Examples 72
3.2.9 File Storage 82
  3.2.9.1 Comparison of TYPE1 and TYPE2 File-based Logs 82
  3.2.9.2 Disk Requirements for File-based Numeric Logs 85
3.3 Building History Logs with Text Files 87
3.3.1 An Example Application of dbConfig 88
  3.3.1.1 Use the History Builder to Configure a Database Template 88
  3.3.1.2 Create Tag Lists for Additional Logs 89
  3.3.1.3 Read History Database into a Text File 90
  3.3.1.4 Generate Like Files 91
  3.3.1.5 Generate a Report for dbConfig Output 93

CONTENTS (continued)

3.3.1.6 Configure the History Database from the Text Files....96
3.3.1.7 Restarting History....97
3.3.2 Dumping and Loading Changes....98
3.3.3 dbConfig....99
3.4 Starting and Stopping Data Collection....103

Chapter 4 - Operation

4.1 Getting Started....105
4.2 When to Use Composite Log Name or Numeric Log Name....105
4.3 Retrieval for Advant Trend Displays....107
4.4 Criteria for Selection for Seamless Retrieval....108
4.5 Interpolation....110

Chapter 5 - Maintenance

5.1 Preventive Maintenance....111
5.2 Database Maintenance Functions....111
5.2.1 History Utilities for GUI-based Database Maintenance....112
5.2.2 hsDBMaint for Command Line Database Maintenance....115
5.2.3 Database Free Space Report....116
5.2.4 Entry Tables Report....118
5.2.4.1 Running Entry Tables Report from the Command Prompt....118
5.2.4.2 Running Entry Tables Report from the History Utility....119
5.2.5 Update Deadband Ratio On-line....121
5.2.6 Extend Tablespace for History Files....122
5.2.6.1 Guidelines for Extending Tablespace....123
5.2.6.2 Extending Tablespace Procedure....123
5.2.7 Maintain Directories for File-based Logs....126
5.2.7.1 Overview....126
5.2.7.2 Directory Maintenance Procedures....128
5.2.8 Reset Object Status States....133

CONTENTS (continued)

5.2.9 Cascade Attributes for Composite Logs 135
5.2.10 Stagger Collection and Storage 136
5.2.11 Display Oracle Statistics 142
5.2.12 Dump a Flat File 142
5.2.13 Bulk Data Export 143
  5.2.13.1 Creating an Input File 147
  5.2.13.2 Bulk Data Export Parameters 149
  5.2.13.3 Running Bulk Data Export from the Command Prompt 151
5.2.14 Create/Drop User Indexes 152
  5.2.14.1 Dropping an Index 153
  5.2.14.2 Creating an Index 153
5.2.15 Dump & Load Configuration 154
  5.2.15.1 Dump Configuration 154
  5.2.15.2 Load Configuration 157
5.2.16 Create or Drop a Product Database 163
5.2.17 Clean History Database 165
5.2.18 Restore or initialize History Logs 166
5.2.19 Reindex Logs 166
5.2.20 Fill Synchronous Entry Tables 166
5.2.21 Purge History Data 167
5.3 Configuring OPC Server Access 169

Chapter 1 Introduction

1.1 About This Book

This book describes Advant® History software version 1.0 (hereafter referred to as History or the History package). This book is intended for application engineers who configure the historical database, and operators and plant supervisors who view, interpret and respond to historical data during runtime. Operators may redefine history objects during runtime if given the proper authority.

As a prerequisite, you should understand distributed automated process control, the hardware and software functionality of an Advant OCS, the platform and operating system that History is operating on, and information storage in databases.

This book is not intended to be the sole source of instruction for History software. It is recommended that those people who use History attend the applicable training courses offered by ABB.

1.1.1 How to Use this Book

Where to Start

If you are an experienced History user, you should refer to Section 1.2.2, What's New in this Release to learn about the functions and features that have been added since the last release. If you are a first-time user, refer to Section 1.2.1, What You Can Do with Advant History to learn about the basic functions and features for this product.

Installing Advant History Software

Advant History software is delivered on a CD with the AdvaSoft 2.0 for Windows NT software set. Refer to the *Advant Workplace Products Administrator's Guide* for instructions on installing this software.

Configuring Advant History

Refer to Chapter 3, Configuration for instructions on how to configure your History database. Section 3.1, Getting Started provides a quick summary of the procedures you use to configure History. This section also refers you to the specific sections where these procedures are described in detail.

Accessing Advant History Data From Other Applications

Chapter 4, Operation provides guidelines for accessing Historical data from external applications such as Advant Trends and Advant Reports.

Maintenance & Administrative Procedures

Chapter 5, Maintenance describes error messages applicable to this software package. It also provides instructions for database maintenance functions such as extending tablespace.

Refer to the *Advant Workplace Products Administrator's Guide* for instructions regarding system administration procedures such as:

- licensing software
- managing users (creating new users, setting up passwords, and configuring user preferences).

1.2 Product Overview

History is an option for the AdvaSoft 2.0 for Windows NT suite of software products.

History supports collection, storage, and retrieval of process data. Other software packages such as Advant Trends and Advant Reports can access the historical data for display in trend graphs, and presentation in reports.

A summary of established functions and features for new users is provided in Section 1.2.1, What You Can Do with Advant History.

A summary of new features and functions for experienced users is provided in Section 1.2.2, What's New in this Release.

The release history of this product is summarized in Section 1.2.3, Release History.

1.2.1 What You Can Do with Advant History

Advant History supports:

- Data Collection
- Data Storage
- Data Retrieval

These functions are performed according to the History Database Configuration. This historical data is available for use in other applications. For instance:

- It can be displayed on trend displays.
- It can be included in reports produced by Advant Reports.
- It is available for custom software applications via SQL (Structured Query Language) and User API (Application Programming Interface).

1.2.1.1 History Database Configuration

You can configure logs on-line using the History Builder, or off-line via the dbConfig utility. The History Builder provides a graphical user interface and log configuration wizard to facilitate history configuration. The dbConfig utility is a command line user interface. This utility lets you use text files to configure a large quantity of logs that have identical configurations that differ only by their respective data sources.

The recommended method for building a history configuration is to create template logs using the History Builder and then use dbConfig to create the remaining logs based on the template logs. See Chapter 3, Configuration.

1.2.1.2 Data Collection

History periodically collects data from a data source using the OPC Data Access Rev 2 specification as the protocol. A common application of history is to periodically store values of process variables such as flows, temperatures, or tank levels. For example, to store the value of a steam flow, your data source is the measured variable of the loop that controls the flow.

Some of the data collection features supported by History include:

- Composite Log Structure
- Calculations
- Data Compaction
- Blocking and Alignment

For details on configuring data collection in History, see Section 3.2.8.2, Calculation Attributes.

Composite Log Structure

Numeric logs are built within a composite log structure. A composite log is a series of connected logs that have a common data source. This structure lets you provide different views of a process variable by configuring a series of logs with different time spans (for instance, high resolution data for a short time span in a primary log, and a lower resolution for a longer time span in a secondary log). This also lets you store data over longer periods without using as much storage capacity.

Data retrieval is easy via the log name (data source). By default, all logs in a composite log take on the data source name as their log name.

For further information regarding composite logs, see Section 3.1.1, The Composite Log Structure.

Calculations

You can configure History to perform calculations on collected data. The types of calculations that you can perform are: AVERAGE, MAXIMUM, MINIMUM, INSTANTANEOUS (no calculation), SUM, SUM_OF_ SQRS, STANDARD DEVIATION, NUM_OF_ VALUES. For further information see Calculation Algorithm in Section 3.2.8.2, Calculation Attributes.

Data Compaction

You can also configure History for data compaction which uses a mathematical compaction algorithm to reduce the amount of data stored for numeric variables. For further information see Section 3.2.8.3, Deadband Attributes.

Blocking and Alignment

Sample Blocking Rate can be used to allocate a buffer in the collection function to hold the blocking rate worth of samples before applying a calculation and (if applicable) deadband algorithms and sending the samples to storage.

The blocking rate can also be used to optimize performance by phasing the writing of historical data to disk. For further information see Sample Blocking Rate in Section 3.2.8.1, Numeric Log Attributes.

1.2.1.3 Data Storage

Log entries are stored directly in files maintained by History. How to configure synchronous numeric logs for file-based storage, and related considerations are described in Section 3.2.9, File Storage.

1.2.1.4 Data Retrieval

Data retrieval refers to the presentation of historical data on operator displays and access of historical data by other applications such as Advant Trends and Advant Reports.

Data Presentation

Data Presentation is supported by Advant trend displays. These displays are simple graphs of process variables versus time. Data presentation is also supported by AdvaInform Display. See Section 4.3, Retrieval for Advant Trend Displays.

History can return the raw data that is actually stored, or interpolated data where entries are calculated at fixed intervals. This is useful when you use deadband and not all the entries are stored. INTERPOLATED will ensure the expected number of values are returned. See Section 4.5, Interpolation.

Advant Reports

History supports Advant Reports by providing historical information for the reports.

1.2.2 What's New in this Release

This is the first release of Advant History. For a brief overview of the functions and features, see Section 1.2.1, What You Can Do with Advant History.

1.2.3 Release History

Table 1-1 lists the major milestones in the development of Advant History software.

Table 1-1. Release History

Version	Description
1.0	This is the initial release of History. For a brief overview of the functions and features, see Section 1.2.1, What You Can Do with Advant History.

1.3 Prerequisites & Requirements

The History software runs on a Windows NT workstation. The computer must have the AdvaSoft for Windows NT installed. General hardware and software requirements for the Windows NT platform with AdvaSoft for Windows NT are described in the *Advant Workplace Products Administrator's Guide*.

1.4 Conventions

The following conventions are used in this book:

- The words in proper names of screen elements (for example, the title in the title bar of a window, the label for a field of a dialog box) are initially capitalized.
- Capital letters are used for the name of a keyboard key if it is labeled on the keyboard. For example, press the ENTER key.
- Lowercase letters are used for the name of a keyboard key that is not labeled on the keyboard. For example, the **space bar**, **comma key**, and so on.
- A plus sign is used to indicate that you must simultaneously press several keys. For example, CTRL+C indicates that you must hold down the CTRL key while pressing the C key.
- The phrase “press and release” is used to indicate that you sequentially press several keys. For example, to close a window, press and release ALT, and then F4. This indicates that you press and release the ALT key, and then you press and release the F4 function key.
- The names of push and toggle buttons are boldfaced. For example, click **OK**.
- The names of menus and menu items are boldfaced. For example, the **File** menu.
  - The following convention is used for menu operations: *menuName* > *menuItem* > *cascadedMenuItem*. For example: choose **File** > **New** > **Type**.
  - The **Start** menu name always refers to the **Start** menu on the Windows NT Task Bar.
- System prompts/messages are shown in the Courier font, and user responses/input are in the boldfaced Courier font. For example, if you enter a value out of range, the following message is displayed:

  ```
  Entered value is not valid. The value must be 0-30.
  ```

  In a tutorial, you may be told to enter the string TIC132 in a field. The string is shown as follows in the procedure: **TIC132**

1.5 Related Documentation

Table 1-2 lists all documentation related to Advant History.

Table 1-2. Related Documentation

Category	Title	Description
System Administration	Advant Workplace Products Administrator's Guide	This book describes how you install the Advant Workplace Products software, including software options, and how you then perform maintenance.
Software	Advant Workplace User's Guide	This book describes everything about the user interface, for example how you present information, navigate and control objects.
	Advant Plant Explorer User's Guide	This book describes how you build hierarchically structured models of a plant or project as well as how you navigate in the structures.
	Advant Graphics User's Guide	This book describes how you build and use graphics in Advant Workplace Products. It includes information about the basic building blocks provided by ABB.
	Advant Alarms User's Guide	This book explains how to use the Advant Alarm and Event functions. It covers alarm and event configuration and presentation.
	Advant Trends User's Guide	This book describes how to use the trend graphs to view historical data during runtime.
	Advant Reports Add-Ins User's Guide	This book describes how to include current and historical information in process reports

Table 1-2. Related Documentation (Continued)

Category	Title	Description
Options	Advant Connectivity Library for Advant Controller 100 Series Reference Manual	This book is a set of pre-defined graphic elements for Advant Controller 100 Series.
	Advant Graphic Library for Process Industry Reference Manual	This book is a set of pre-defined graphics used by the process industry.
	Advant OPC Server for Advant Fieldbus 100 User's Guide	This guide describes the Advant OPC Server for Advant Fieldbus 100 (Advant OPC Server for AF100), which is a standard OPC interface for Advant Fieldbus 100.
	Advant History User's Guide	This book describes the Advant History software.
Other	Advant Workplace Products for Operators Product Guide	This book is a presentation of the software and software options in Advant Workplace Products for Operators. It also includes hardware and software requirements and specifications.
	Advant Workplace Engineering Methods Reference Manual	This book is intended as a guide for plant engineering using Advant Workplace Products software together with Advant Controller 100 Series.
	AdvaSoft 2.0 for Windows NT User's Guide	This is the operators manual and contains information on configuration and use of the AdvaSoft 2.0 for Windows NT.

1.6 Terminology

Term	Description
API	A set of subroutines that provide a means to establish remote process-to-process communication.
Attribute	The public data of an object. The result of the application and the configured data is commonly stored in an attribute.
Controller	This refers to the Advant Controller which handles I/O and executes control logic in the Advant OCS.
Control Network	A network connecting different types of Advant stations/controllers used to supervise and control a specific part of the plant.
Command	Operation requests to an object. As a result, the object performs the requested action.
External Node	An external node is a node that is not part of the Advant OCS.
History Object	History objects support historical execution. Examples of history objects are logs and log managers. History objects are built with the History Builder.
Log Entries	The collected values that have been stored in history
Aspect	An aspect is a category of data associated with an object, for instance a TrendDisplay aspect may be associated with a plant area object. Aspects are opened to give access to their data.
Object	A real plant object, for example a valve, is represented by an **ABB Object** in the Advant automation system. An ABB Object contains data describing the different **aspects** of it.

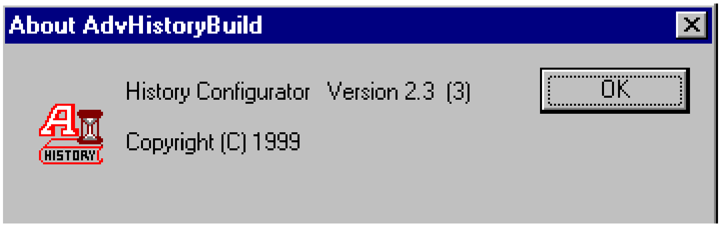
Term	**Description**
Log	The basic storage unit in History. A log represents the stored values and configuration information of the log
Composite Log	A collection of logs that store data for a single data source. Typically, the logs are in a hierarchical fashion. A data source is an object.atrribute combination. Therefore there can be multiple composite logs for a single data source (i.e. PID.SETPT)

Chapter 2 Installation

Advant History is a part of the AdvaSoft 2.0 for Windows NT software package. The whole package is installed according to documentation included in the *Advant Workplace Products Administrator's Guide*.

2.1 Product Verification

Go to the menu and choose **Help>About Advant History**. This displays an information box with such information as software version and licensing, Figure 2-1. To close the box, click **OK**.



History Configurator Version 2.3 (3)

Copyright (C) 1999

Figure 2-1. About Advant History Builder

Chapter 3 Configuration

3.1 Getting Started

You can configure logs on-line with the History Builder, or off-line via the dbConfig utility. The History Builder provides a graphical user interface and log configuration wizard to facilitate history configuration. The dbConfig utility is a command line user interface. This utility lets you use text files to configure a large quantity of logs that have identical configurations that differ only by their respective data sources.

It is recommended that you start by using the History Builder to create a template log for each basic configuration required by your application, and then use those template logs with the dbConfig utility to create the vast majority of logs.

The recommended work-flow is briefly outlined below.

Learning the Basics

The basic unit of storage in History is the composite log. For a quick overview, see Section 3.1.1, The Composite Log Structure.

The History Builder user interface is based on the History Builder window. For a quick introduction, refer to Section 3.1.2, User Interface.

Using Advanced Data Collection and Storage Features

The log configuration wizard helps you configure a basic numeric log; however, some of the advanced data collection and storage features such as data compaction and calculation algorithms that directly affect CPU load and disk usage are not covered by the wizard. If you anticipate needing to use such advanced features, you should consult the following sections before you actually begin configuring logs:

- Section 3.2.8.2, Calculation Attributes
- Section 3.2.8.3, Deadband Attributes
- Section 3.2.9, File Storage

Also refer to Section 3.2.8.5, Data Collection Examples. These examples provide guidelines and recommendations for common data collection/storage applications:

- Example 1, Storing Instantaneous Values, No Compaction
- Example 2 - Storing Instantaneous Values Using Compaction
- Example 3 - Storing Calculated Values in a Primary Log
- Example 4 - Storing Calculated Values in Primary and Secondary Logs
- Example 5- Storing Calculated Values with Compaction

Allocating Disk Space for File-based Logs

You may need to expand storage capacity for file-based logs. For file-based storage (Storage Type = TYPE1 or TYPE2), you must specify where to store the file-based logs. This involves specifying the directory (or disk), and how much of the disk is to be allocated to History.

Allocation of disk space must be done before any file-based logs can be created. You can perform this function on-line via the Directory Maintenance History Utility as described in Section 5.2.7, Maintain Directories for File-based Logs, or off-line using the hsDBMaint functions as described in Section 5.2.2, hsDBMaint for Command Line Database Maintenance.

Starting the History Builder

To start the History Builder, see Section 3.1.3, Application Start-up.

Using the History Builder to Create Template Logs

To create the basic templates for each composite log type required by your application, see Section 3.2, Configuring Composite Logs via the History Builder.

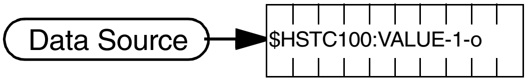
Completing the Database Configuration using dbConfig

To finish the remainder of your History database configuration using the dbConfig utility, see Section 3.3, Building History Logs with Text Files.

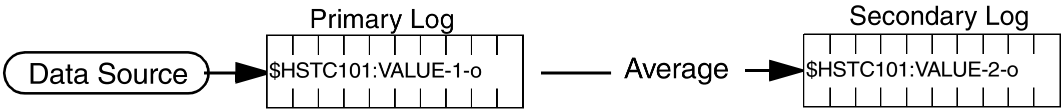
History must be re-started in order for logs created using dbConfig to collect and store data.

3.1.1 The Composite Log Structure

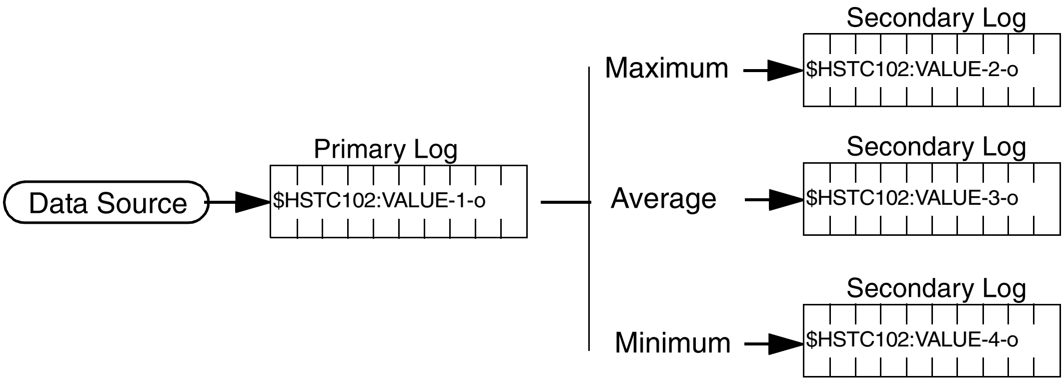
Numeric logs are built within the composite log structure. A composite log can be as simple as a single *primary* log (a log that is connected directly to a data source), or it can have multiple primary and *secondary* logs (secondary logs get their data either from a primary log, or another secondary log). Some example composite log configurations are illustrated in Figure 3-1.



Example- Simple Primary Log



Example- Primary Log w/Secondary Log Storing Calculated Average



Example- Primary Log w/ Multiple Secondary Logs Storing Calculated Values

Figure 3-1. Example Composite Log Configurations

3.1.2 User Interface

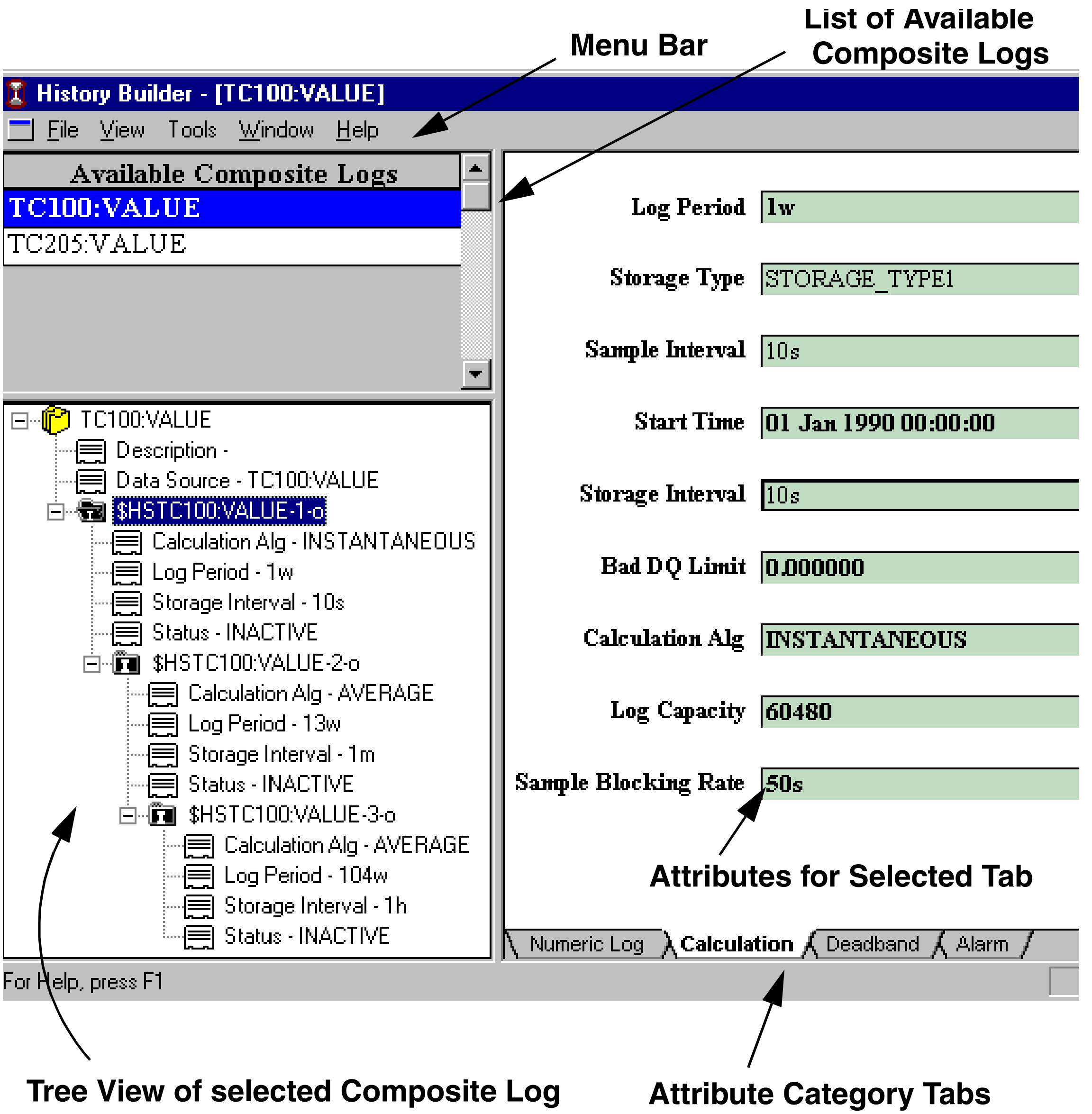
The user interface for History is based on Windows NT and the Advant Workplace. Refer to the applicable Windows NT documentation, and to the *Advant Workplace Products Administrator's Guide* for details.

The History Builder window, Figure 3-2, provides a graphical user interface for configuring History logs. Functionally, the History Builder window is divided into four basic sections: the Available Composite Logs List, the Tree View for a selected composite log, the Log Attribute Tabs, and the Main Menu Bar.

3.1.2.1 Available Composite Logs List

The Available Composite Logs list, Figure 3-2, shows all composite logs that have been configured and are stored in the History database. Use this list to either select an existing log for further configuration, or to add a new composite log.

When you first open the History Builder window, the Available Composite Log list is empty. To retrieve the available composite logs, right click anywhere in the list, and then choose **Get Full Log List** from the context menu. When you select a composite log from this list, the corresponding log hierarchy is presented in the Tree View.



TC100:VALUE
TC205:VALUE

- TC100:VALUE
  - Description -
  - Data Source - TC100:VALUE
  - $HSTC100:VALUE-1-o
    - Calculation Alg - INSTANTANEOUS
    - Log Period - 1w
    - Storage Interval - 10s
    - Status - INACTIVE
    - $HSTC100:VALUE-2-o
      - Calculation Alg - AVERAGE
      - Log Period - 13w
      - Storage Interval - 1m
      - Status - INACTIVE
      - $HSTC100:VALUE-3-o
        - Calculation Alg - AVERAGE
        - Log Period - 104w
        - Storage Interval - 1h
        - Status - INACTIVE

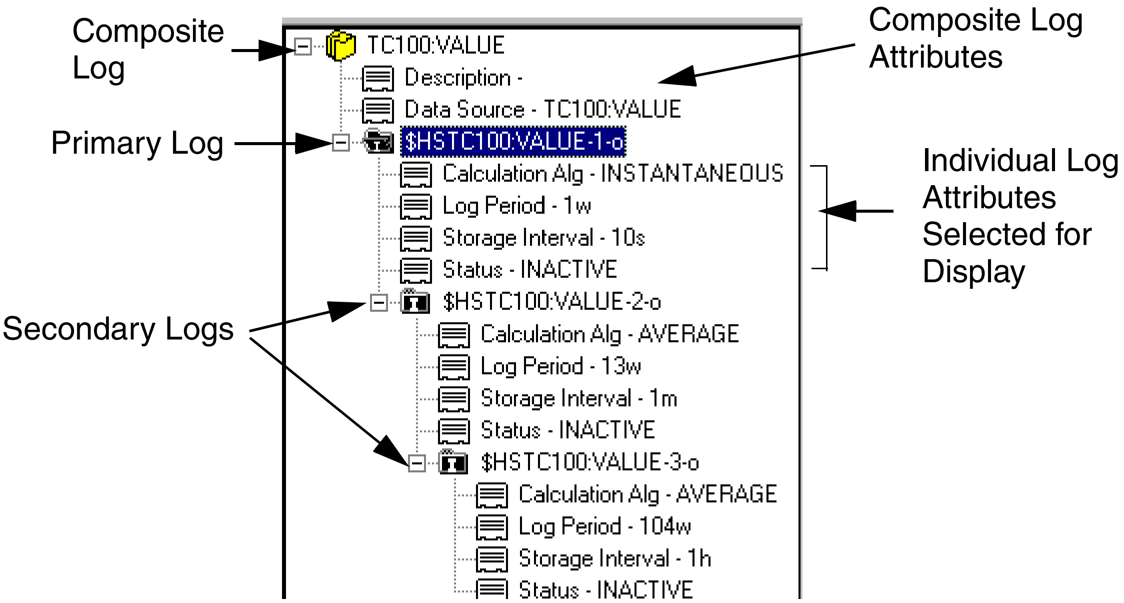
Log Period	1w
Storage Type	STORAGE_TYPE1
Sample Interval	10s
Start Time	01 Jan 1990 00:00:00
Storage Interval	10s
Bad DQ Limit	0.000000
Calculation Alg	INSTANTANEOUS
Log Capacity	60480
Sample Blocking Rate	50s

Figure 3-2. History Builder Window

3.1.2.2 Tree View

The Tree View shows the hierarchy for any composite log that you select from the Available Composite Logs List, Figure 3-3. This hierarchy includes the composite log name, composite log data source and description attributes (shown on an optional basis), plus the composite log's component primary and secondary logs, and their respective log attributes. You can select up to five attributes to show in the composite log hierarchy. The same set of attributes are shown for all primary and secondary logs.

A composite log can have a maximum of 16 primary and secondary logs. Each primary and secondary log is represented by a folder in the hierarchy. The folders are color-coded to indicate the log's configuration and runtime status.



- TC100:VALUE
  - Description -
  - Data Source - TC100:VALUE
  - $HSTC100:VALUE-1-o
    - Calculation Alg - INSTANTANEOUS
    - Log Period - 1w
    - Storage Interval - 10s
    - Status - INACTIVE
    - $HSTC100:VALUE-2-o
      - Calculation Alg - AVERAGE
      - Log Period - 13w
      - Storage Interval - 1m
      - Status - INACTIVE
      - $HSTC100:VALUE-3-o
        - Calculation Alg - AVERAGE
        - Log Period - 104w
        - Storage Interval - 1h
        - Status - INACTIVE

Figure 3-3. Tree View

Color and Symbol Key for Tree View Objects

The color/symbol key is shown in Figure 3-4. To show this key, right click in the Tree View and choose **Show Color Key**.

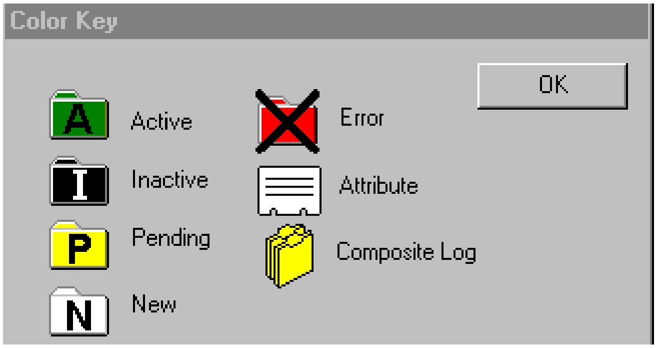


Figure 3-4. Color Key

Description of Colors:

- White: Log is new, it has not been saved, there are no errors in the configuration
- Red: There is an error in the log configuration
- Green: The log is collecting and storing data
- Yellow: The log has been activated and is awaiting its alignment time to start collecting
- Black: The log is inactive

Tree View Context Menu

Most of the work you do in the History Builder window is via Tree View's right-click menu. This menu lets you:

- Customize the tree view. This includes choosing whether or not to show composite log attributes, and selecting which numeric log attributes to show in the tree. For details, see Section 3.2.2, Customizing the Tree View.
- Add Logs in the Tree View. For details, see Section 3.2.4, Adding Numeric Logs to the Composite Log.
- Access Log Attributes

  For any log attribute that is shown in the tree view, you can display a log attribute dialog by double-clicking on the attribute name. This is described further in Section 3.2.6.1, Configuring Attributes from the Tree View.

  To access the complete list of attributes for a log, click on the log name. This makes the selected log's attributes available via the Log Attribute Tabs on the right side of the History Builder window.
- Starting and Stopping Logs. See Section 3.2.7, Starting/Stopping Logs.

3.1.2.3 Log Attribute Tabs

When you select an individual log in the Tree View, that log's attributes are available via the log attribute tabs on the right side of the History Builder window. The log attributes are organized under the following four tabs:

- Numeric Log Attributes
- Calculation Attributes
- Deadband Attributes
- Alarm Attributes

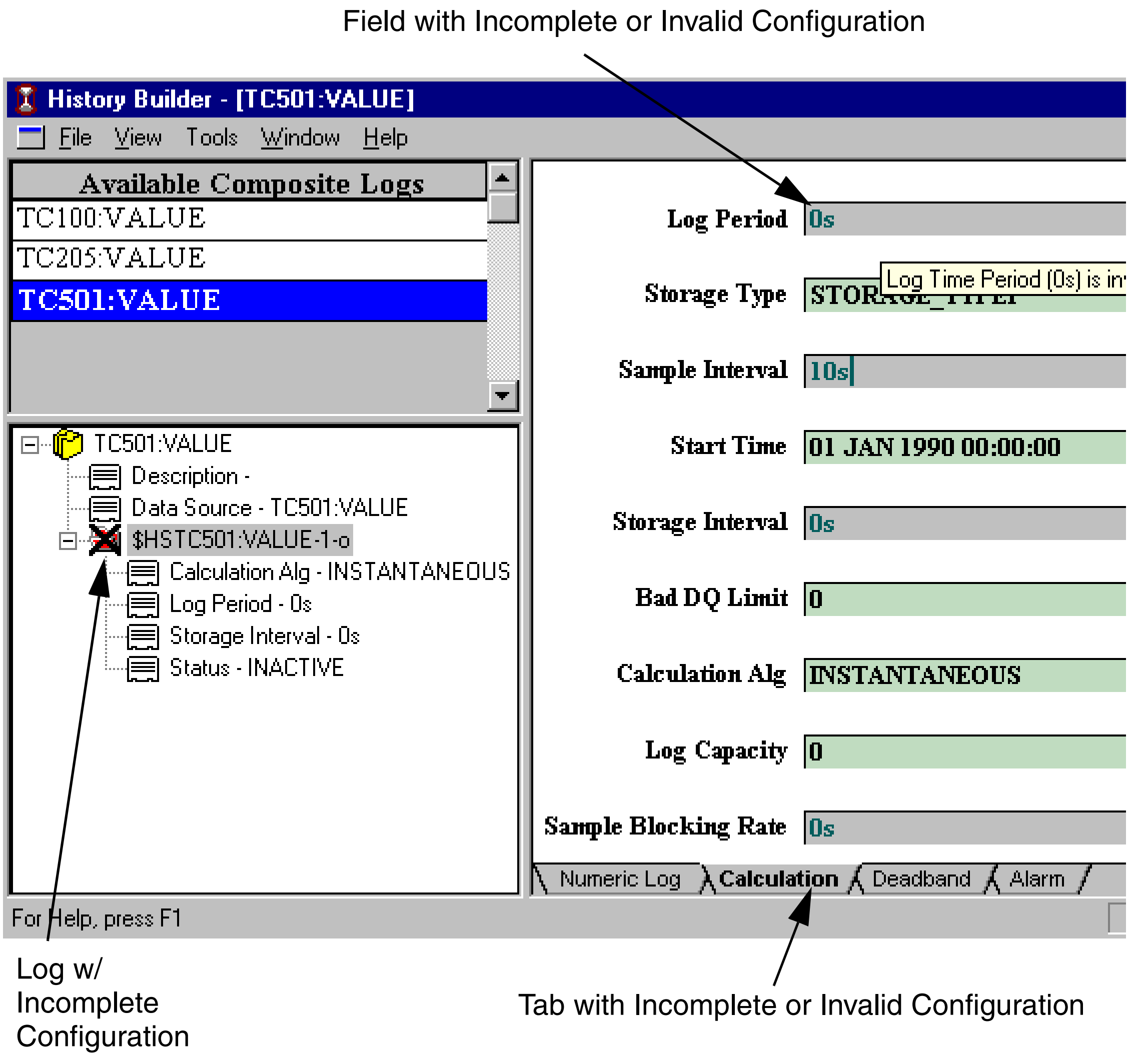
3.1.2.4 Configuration Status Indication

Incomplete or invalid configuration of one or more log attributes is indicated on three levels as shown in Figure 3-5, and described below:

- The log's folder in the Tree View is red and crossed out.
- The tab where the attribute is listed is red.
- The attribute's field on the tab is red.

To get details regarding an attribute whose configuration is incomplete, either:

- hold the mouse pointer over the field. This displays a pop-up message.
- click in the field. This displays a message in the bottom left window bar.



TC100:VALUE
TC205:VALUE
TC501:VALUE

- TC501:VALUE
  - Description -
  - Data Source - TC501:VALUE
  - $HSTC501:VALUE-1-o
    - Calculation Alg - INSTANTANEOUS
    - Log Period - 0s
    - Storage Interval - 0s
    - Status - INACTIVE

Log Period	0s
Storage Type	STOR
Sample Interval	10s
Start Time	01 JAN 1990 00:00:00
Storage Interval	0s
Bad DQ Limit	0
Calculation Alg	INSTANTANEOUS
Log Capacity	0
Sample Blocking Rate	0s

Log Time Period (0s) is in

For Help, press F1

Field with Incomplete or Invalid Configuration

Log w/
Incomplete
Configuration

Tab with Incomplete or Invalid Configuration

Figure 3-5. Indicators for Incomplete Configuration

3.1.2.5 Main Menu Bar

The menu items available via the menu bar are determined by the method you use to start the History Builder. If you start the History Builder independent of the Plant Explorer, main menu bar provides the following menus: File, View, Tools, Window, and Help.

File

New Window	Opens a new History Builder window
Save Composite Log	Save the current composite log. For an example, see Saving Your Work in Section 3.2.4.1, Simple Numeric Logs for Synchronous Process Data.
Save Composite Log As	Create a new composite log which is a duplicate of the current composite log, except that it has a different data source. For an example, see Creating a Composite Log by Copying an Existing One in Section 3.2.4.1, Simple Numeric Logs for Synchronous Process Data
Delete Composite Log	Delete the current composite log.
Print	The Print, Print Preview, Print Setup, and Recent Composite Logs menu items are not operational for this release..
Print Preview	
Print Setup	
Recent Composite Logs	
Exit	Exit the History Builder.

View

Toolbar	Show/hide the toolbar.
Status Bar	Show/hide the status bar.

Tools

History Utilities	Open the History Utilities window. See Section 5.2, Database Maintenance Functions.

Window

Cascade Tile Arrange Icons	Standard Windows NT functions.

Help

Help Topics	Open On-line help for History Builder.
About History Builder	Display About History Builder message box with version and licensing information.

If you start the History Builder independent of the Plant Explorer, main menu bar provides the following menus: File, View, Tools, Window, and Help.

File

Save Composite Log	Save the current composite log. For an example, see Saving Your Work in Section 3.2.4.1, Simple Numeric Logs for Synchronous Process Data.
Save Composite Log As	Create a new composite log which is a duplicate of the current composite log, except that it has a different data source. For an example, see Creating a Composite Log by Copying an Existing One in Section 3.2.4.1, Simple Numeric Logs for Synchronous Process Data
Delete Composite Log	Delete the current composite log.
Print	
Print Preview	The Print, Print Preview, Print Setup, and Recent Composite Logs menu items are not operational for this release.
Print Set	
Recent Composite Logs	
Exit	Exit the History Builder.

View

Log Configuration Wizard	Start Log Configuration Wizard. See Configuring Log Attributes with the Log Configuration Wizard.
Status Bar	Show/hide the status bar.
Split	
Expand Tree	Expand the tree to show all numeric logs and their respective attributes.
Collapse Tree	Collapse tree to show just the Composite Log.
Refresh Tree	
Tree Options	Display the Options dialog. See Section 3.2.2.1, Select Which Attributes to Show for Primary/Secondary Logs.

Appearance Options	
Show Color Key	Show the color key. See Color and Symbol Key for Tree View Objects.

Tools

History Utilities	Open the History Utilities window. See Section 5.2, Database Maintenance Functions.

Help

Help Topics	Open On-line help for History Builder.
About History Builder	Display About History Builder message box with version and licensing information.

3.1.3 Application Start-up

To open the HistoryLogConfiguration aspect for a particular plant object, you can either use the Windows NT task bar, or the Plant Explorer. To use the Plant Explorer, navigate to the HistoryLogConfiguration aspect for that object, and then either double-click, or choose **Open** from the right-click menu. Refer to the *Advant Plant Explorer User's Guide* for details.

To start the History Builder, from the Windows NT task bar choose:
Advant > History > HistoryBuilder, Figure 3-6.

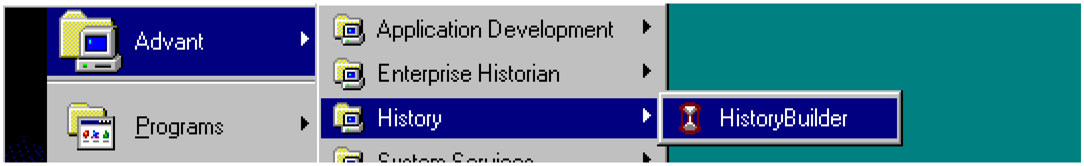


Figure 3-6. Starting the History Builder

3.2 Configuring Composite Logs via the History Builder

To get started configuring logs with the History Builder:

1. When the History Builder is first opened, the Available Composite Logs List is empty. You are not required to show existing composite logs. You may find it easier to add new composite logs without showing the existing logs. If you prefer to see all existing logs, see Section 3.2.1, Getting the Full Log List.
2. If you want to change any of the default presentation settings for the History Builder Tree View, see Section 3.2.2, Customizing the Tree View.
3. Using the Available Composite Logs List, either select an existing composite log to work with, or create a new composite log (Section 3.2.3, Adding a Composite Log. The corresponding log hierarchy is presented in the Tree View.
4. Add primary and secondary logs to the composite log via the Tree View. See Section 3.2.4, Adding Numeric Logs to the Composite Log.
5. Adjust numeric log configurations if necessary. See Section 3.2.6, Adjusting Numeric Log Attributes.

Additional operations that you can perform using the History Builder window are:

- Deleting Logs
- Starting/Stopping Logs either on an individual numeric log basis, or on a composite log basis.

3.2.1 Getting the Full Log List

When you first open the History Builder window, the Available Composite Log list is empty. To retrieve the available composite logs, right click anywhere in the list, and then choose **Get Full Log List** from the context menu, Figure 3-7.

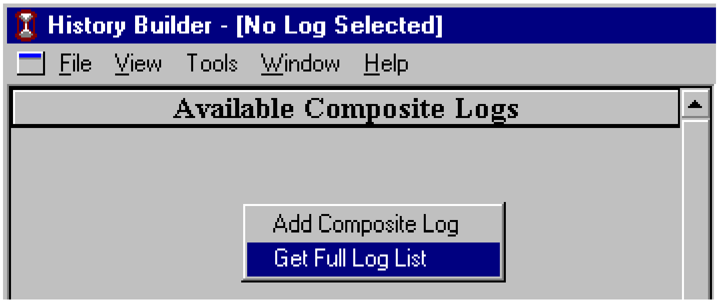


Figure 3-7. Retrieving the Existing Composite Logs

When you select a composite log from this list, the corresponding log hierarchy is presented in the Tree View. If you want to change the default presentation for the Tree View, see Section 3.2.2, Customizing the Tree View. For instructions on how to add a composite log to this list, see Section 3.2.3, Adding a Composite Log.

3.2.2 Customizing the Tree View

You can:

- Select Which Attributes to Show for Primary/Secondary Logs
- Show or Hide the Composite Log Attributes
- Expand the Tree View

3.2.2.1 Select Which Attributes to Show for Primary/Secondary Logs

You can select up to five log attributes to show under each log's individual folder. The same attributes are shown for all primary and secondary logs in the composite log. By default, no attributes are shown. To change the default selection, right click in the Tree View and choose **Tree Options** from the context menu. This displays the Tree Options dialog, Figure 3-8.

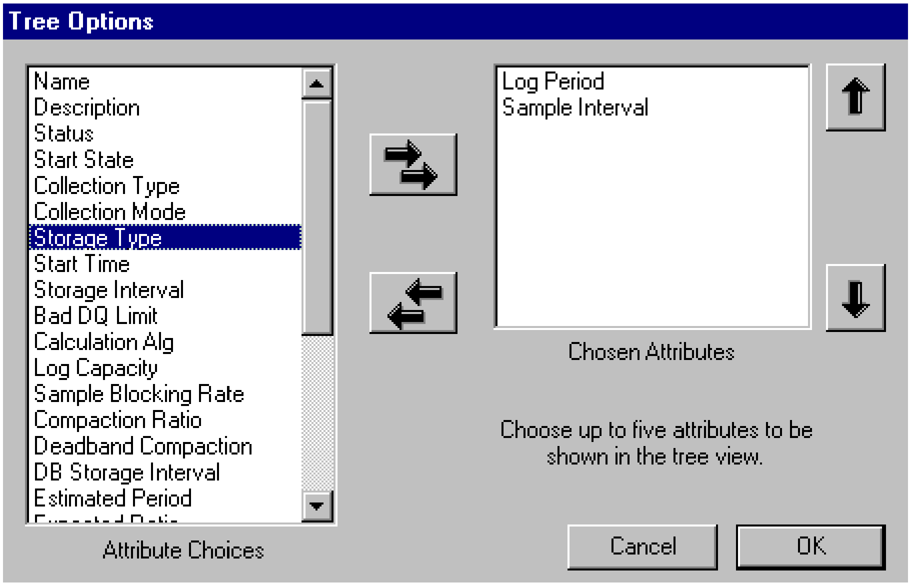


Figure 3-8. Log Configuration, Tree Options Dialog

From the Attribute Choices list, select the attributes that you want to show in the Tree View, and click the > button to put the selected attributes in the Chosen Attributes list.

You can reverse a selection by moving an attribute from the Chosen Attributes list back to the Attribute Choices list, using the < button. If none are selected, clicking < removes all from the chosen attributes list.

Click **OK** when you are finished.

Attribute Choices

This is a list of all available attributes that can be chosen for display in the Tree View.

Chosen Attributes

This is a list of the attributes that are currently chosen for display in the Tree View.

3.2.2.2 Show or Hide the Composite Log Attributes

The composite log attributes (Data Source and Description) are shown in the composite log hierarchy by default, Figure 3-3. You may alternately hide or show these attributes by right-clicking and choosing **Show CL Attributes**.

3.2.2.3 Expand the Tree View

The **Expand Tree** menu item in the right-click menu completely expands the tree view to show all primary and secondary logs, and their respective log attributes.

3.2.3 Adding a Composite Log

You can either add a new (empty) composite log, or make a copy of an existing one (by saving under a new name). When you copy an existing log, you get the existing log's numeric log hierarchy and log attribute definitions. Only the data source is changed.

Adding a New Composite Log

To add a composite log, right click in the Available Composite Logs List, and choose **Add Composite Log** from the context menu, Figure 3-9. This displays the Add Composite Log dialog.

NOTE

If you are configuring another composite log and have not saved your work yet, you will be prompted to save that log before you add a new one.

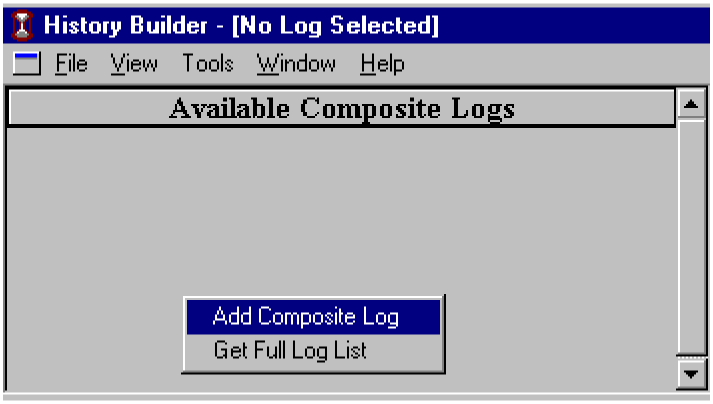
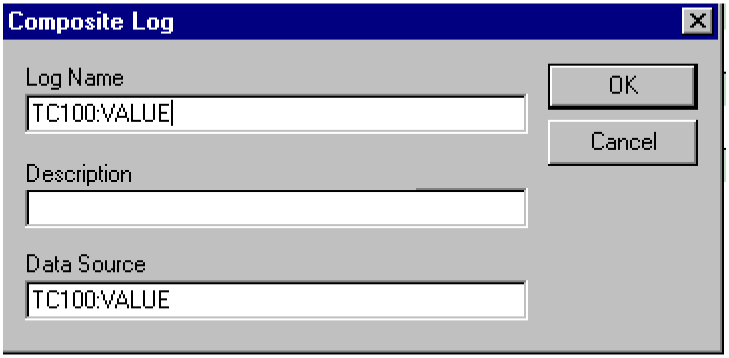


Figure 3-9. Adding a Composite Log

Enter the Composite Log Name and Description. The Data Source is entered automatically as you enter the log name, Figure 3-10.



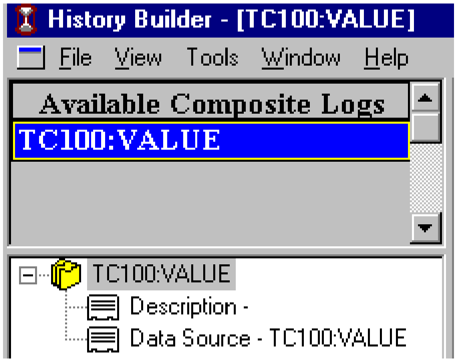
Log Name: TC100:VALUE

Description:

Data Source: TC100:VALUE

Figure 3-10. Add Composite Log Dialog

Click **OK** when you are finished. This displays the starting hierarchy for the new composite log, Figure 3-11.Whether or not the composite log description and data source are shown depends on whether you choose to show them or not. See Section 3.2.2.2, Show or Hide the Composite Log Attributes.



Available Composite Logs
TC100:VALUE

TC100:VALUE
Description -
Data Source - TC100:VALUE

Figure 3-11. New Composite Log Displayed in Tree View

Composite Log Name

The composite log name is the basis for naming all primary and secondary logs in the composite log (See Numeric Log Name in Section 3.2.8.1, Numeric Log Attributes).

The syntax for entering this name depends on the collection method for the log as determined by the Collection Type log attribute.

For applications where you are using History to collect data from an OPC server, the Collection Type is set to **OPC**. In this case the composite log name must specify the OPC tag that will be the Data Source for this composite log.

A number of pre-determined OPC servers are supported. These are defined in a configuration file as described in Section 5.3, Configuring OPC Server Access. Typically, data collection is via the Advant OPC server. In this case the correct format is: *object:attribute*. Use all uppercase letters when you enter the attribute name. For example: **`TC100:VALUE`**

The text string can be up to 65 characters maximum. All characters allowed except: ! ? * % - $

The period character (.) is allowed, but it is strongly recommended that you do not use it unless absolutely necessary. If you do not match the log name with the data source and attribute, you will not be able to retrieve log data using the name of the process object.

Description

This is an optional text string for documentation purposes only. The Description can be up to 40 characters maximum. All characters allowed.

Data Source

The data source is filled in automatically when you enter the log name. The data source can be up to 64 characters.

3.2.4 Adding Numeric Logs to the Composite Log

A composite log can have a maximum of 16 primary and secondary logs. This section shows you how to build the following composite log structures:

- Simple Numeric Logs for Synchronous Process Data
- Secondary Logs for Different Resolution/Time Span Views
- Primary and Secondary Logs with Calculations

More example composite log structures are provided in Section 3.2.8.5, Data Collection Examples.

3.2.4.1 Simple Numeric Logs for Synchronous Process Data

The most basic numeric log structure is a simple primary log that collects real time process data in a synchronous (periodic) mode. A primary log gets its data directly from the Data Source. A composite log may have more than one primary log.

Start by either adding a new composite log as described in Section 3.2.3, Adding a Composite Log, or by selecting an existing log from the Available Composite Logs List.

To add a primary log:

1. Right-click on the composite log name and choose **Add Numeric Log** from the context menu, Figure 3-12.

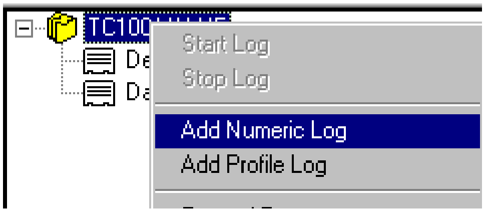


Figure 3-12. Adding a Numeric Log to a Composite Log

The primary log is placed directly under the composite log. You can expand (show) and collapse (hide) the composite log hierarchy by clicking the composite log's (+/-) symbol, Figure 3-13.

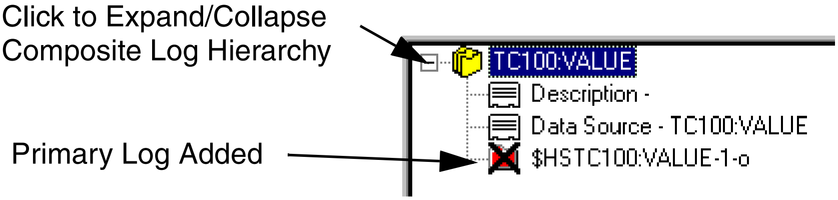

Click to Expand/Collapse Composite Log Hierarchy
Primary Log Added
TC100:VALUE
Description -
Data Source - TC100:VALUE
$HSTC100:VALUE-1-o

Figure 3-13. Primary Log Added Under Composite Log's Data Source

Configuring Log Attributes with the Log Configuration Wizard

The incomplete configuration status is indicated by the red folder. You can configure the log attributes either via the Log Configuration wizard, by double-clicking an attribute in the Tree View, or via the log attribute tabs. The following procedure shows how to use the wizard:

1. Select the log, right click and choose **Log Configuration Wizard** from the popup menu, Figure 3-14.

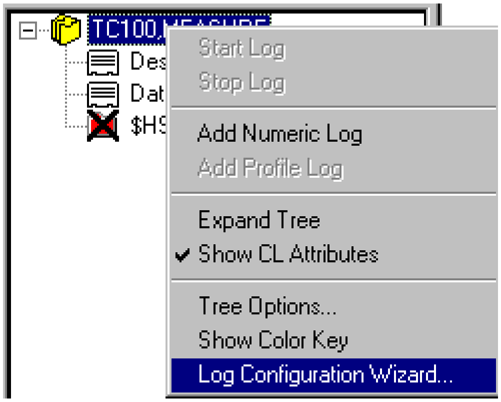
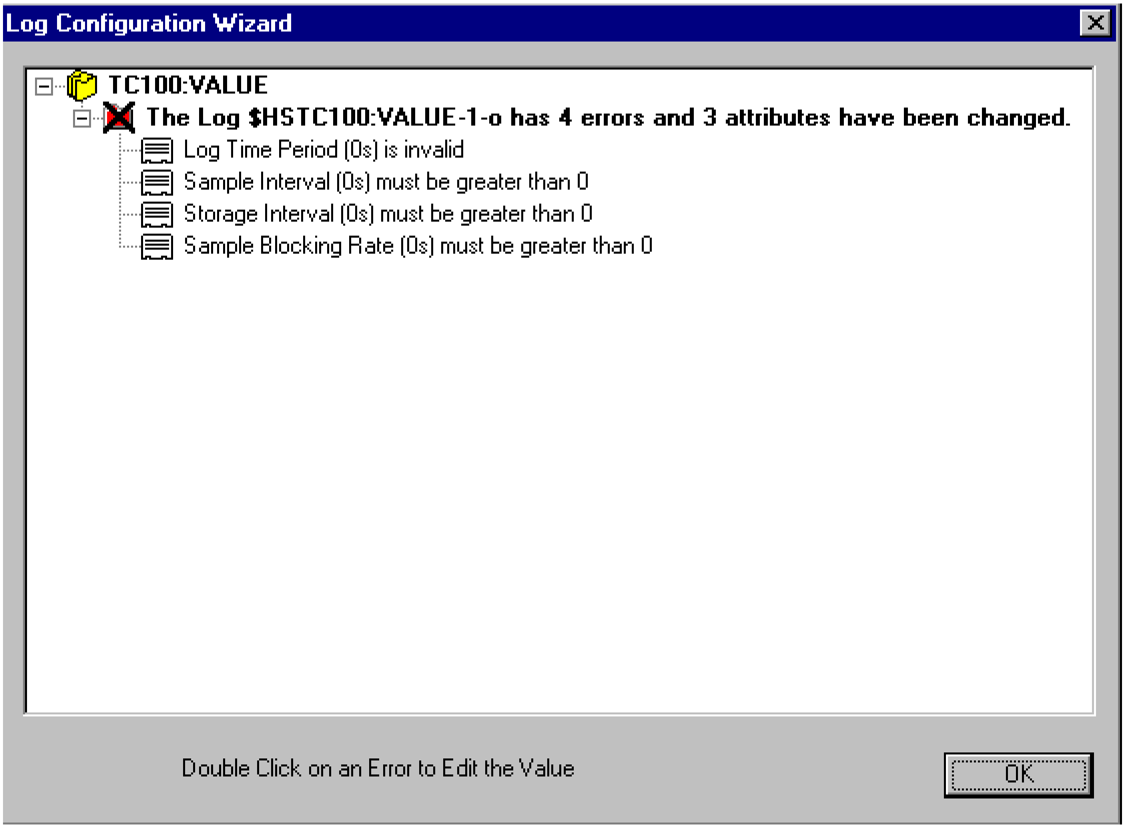


Figure 3-14. Starting the Log Configuration Wizard

This displays the Log Configuration Wizard, Figure 3-15. The wizard lists all attributes that MUST be configured to make this log's configuration valid. Optional attributes (attributes that may be configured on an optional basis) are not shown.

In this example, the following attributes must be configured: Log Period, Sample Interval, Storage Interval, and Sample Blocking Rate. For a detailed description of these and other Calculation attributes, see Section 3.2.8.2, Calculation Attributes.



Log Configuration Wizard

TC100:VALUE

The Log $HSTC100:VALUE-1-o has 4 errors and 3 attributes have been changed.

Log Time Period (0s) is invalid

Sample Interval (0s) must be greater than 0

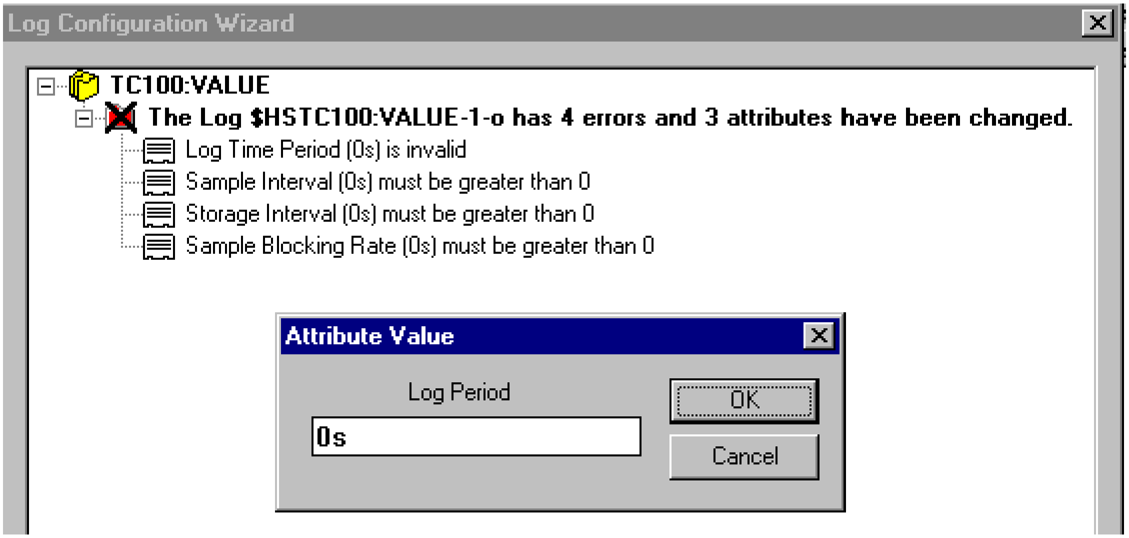
Storage Interval (0s) must be greater than 0

Sample Blocking Rate (0s) must be greater than 0

Double Click on an Error to Edit the Value

Figure 3-15. Example, Log Configuration Wizard

2. Double-click the first attribute (Log Period). This displays the Attribute Value dialog, Figure 3-16.

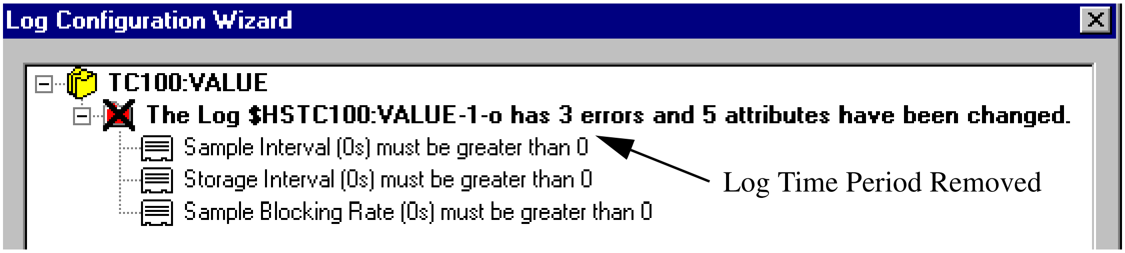


The Log $HSTC100:VALUE-1-o has 4 errors and 3 attributes have been changed.
Log Time Period (0s) is invalid
Sample Interval (0s) must be greater than 0
Storage Interval (0s) must be greater than 0
Sample Blocking Rate (0s) must be greater than 0

Log Period: 0s

Figure 3-16. Attribute Value Dialog

3. Enter a value for Log Period, for example **13w** (13 weeks), then click **OK**. This removes the log time period attribute from the list, Figure 3-17.

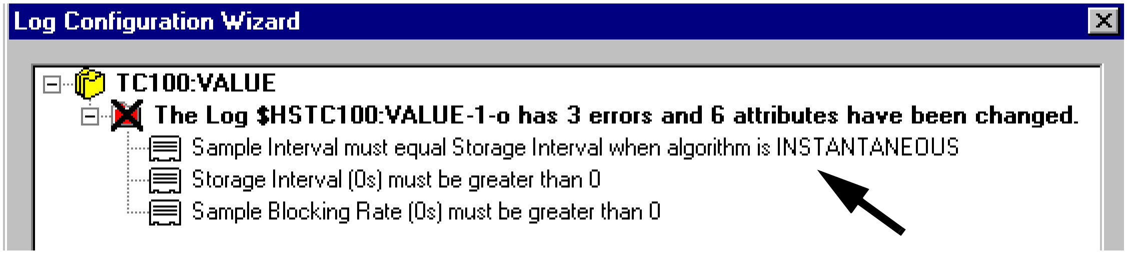


The Log $HSTC100:VALUE-1-o has 3 errors and 5 attributes have been changed.
Sample Interval (0s) must be greater than 0
Storage Interval (0s) must be greater than 0
Sample Blocking Rate (0s) must be greater than 0

Figure 3-17. Configured Attribute Removed from List

4. Repeat steps 2 & 3 for the next attribute in the list (Sample Interval). Enter a valid value for Sample Interval. For example, **10s** = 10 seconds.

Notice that when you specify a value for Sample Interval, the attribute is not removed from the list. Instead, the wizard tells you the Storage Interval value must be equal to Sample Interval when the Calculation Algorithm is Instantaneous, Figure 3-18.

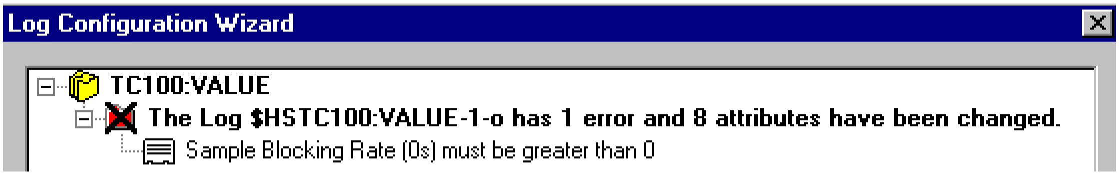


TC100:VALUE
The Log $HSTC100:VALUE-1-o has 3 errors and 6 attributes have been changed.
Sample Interval must equal Storage Interval when algorithm is INSTANTANEOUS
Storage Interval (0s) must be greater than 0
Sample Blocking Rate (0s) must be greater than 0

Figure 3-18. Wizard Providing Guidelines

This is because these three attributes are related. How you configure one is dependent upon how you configure the others. Therefore, even though you entered a valid value for Sample Interval, the wizard does not recognize it as valid until it sees a compatible value for Storage Interval.

5. Double-click on Storage Interval, enter a value equal to the one you entered for Sample Interval, then click **OK**. Now that both Sample Interval and Storage Interval are properly configured, both are removed from the list, Figure 3-19.



TC100:VALUE
The Log $HSTC100:VALUE-1-o has 1 error and 8 attributes have been changed.
Sample Blocking Rate (0s) must be greater than 0

Figure 3-19. More Configured Attributes Removed from the List

6. Complete the configuration of the primary log by configuring a valid value for Sample Blocking Rate (for example 50s). You can be sure the log is configured properly when there are no attributes remaining in the wizard's list, Figure 3-20.

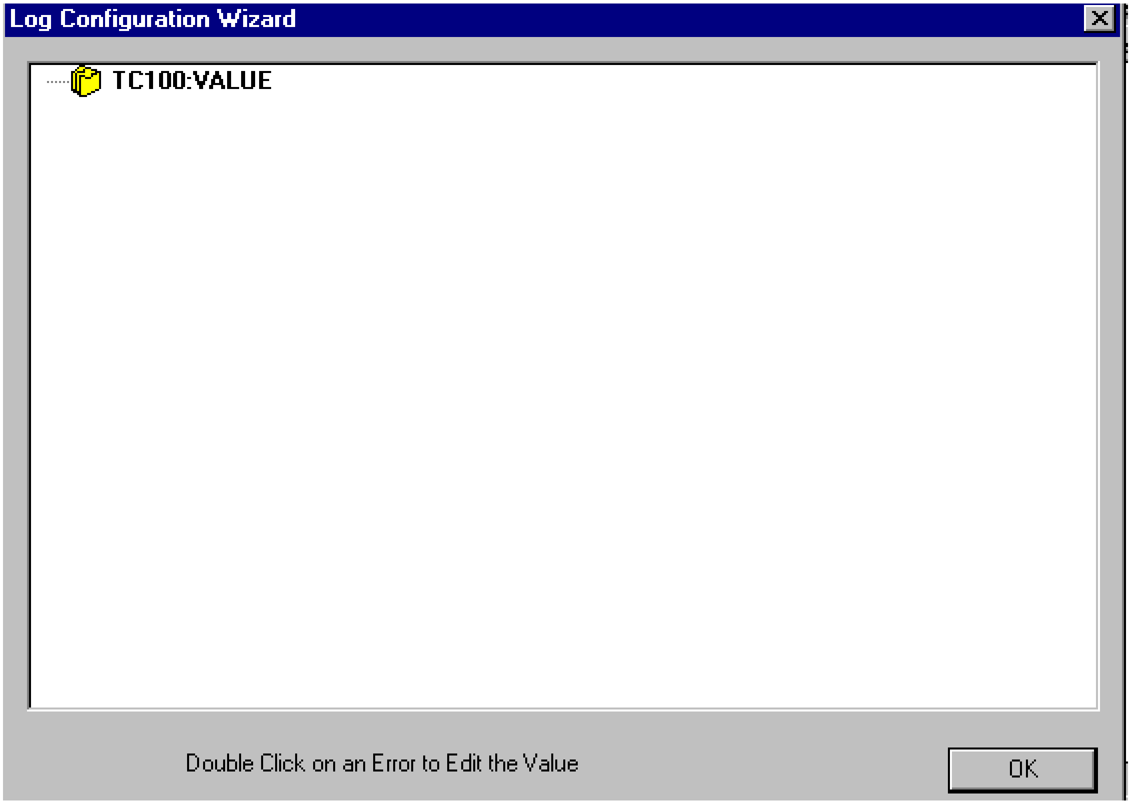


Figure 3-20. Configuration Via Wizard Completed

7. Click **OK** to close the Wizard.

You can retain the defaults for the remaining attributes, or you can configure additional attributes to meet the requirements of your application. If you want to configure additional attributes, refer to Section 3.2.8.1, Numeric Log Attributes for details.

Saving Your Work

Once you have successfully configured all mandatory attributes, the primary log's folder changes to white, and is marked with a large, bold **N,** Figure 3-21.

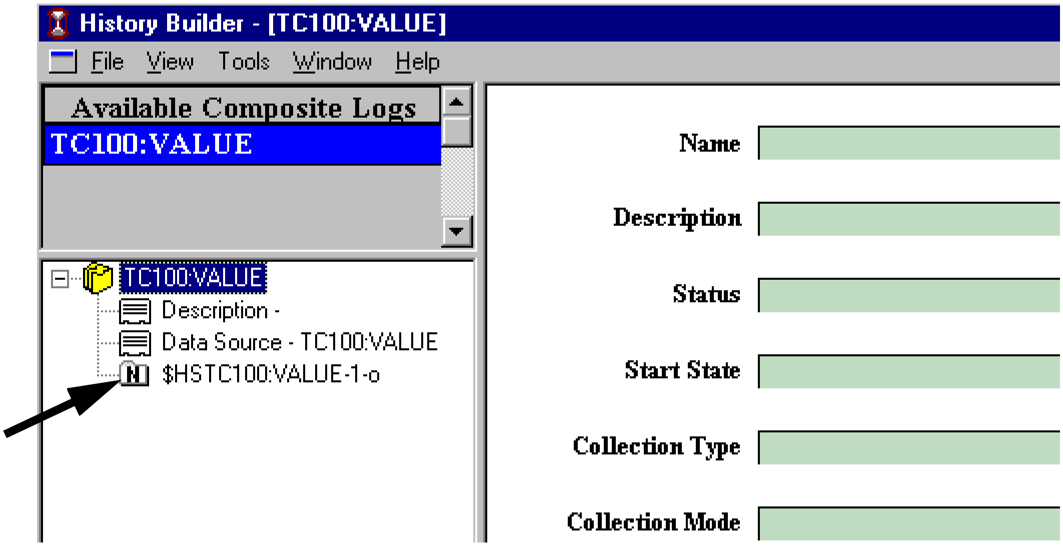


Figure 3-21. Valid Configuration Indication

This means the log is new and has not been saved yet. You can configure additional primary and secondary logs for this composite log without saving; however, you must save before leaving this composite log. To save your work choose **File>Save Composite Log** from the main menu bar, Figure 3-22.

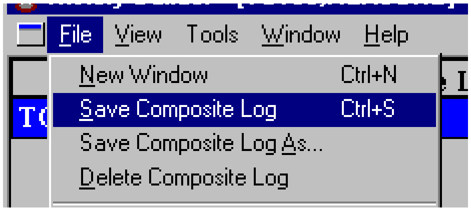
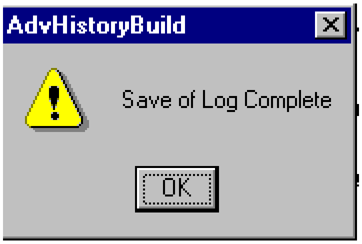


Figure 3-22. Saving a Composite Log

When the save is complete, click **OK** to confirm, Figure 3-23.

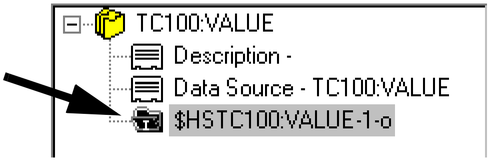


AdvHistoryBuild

Save of Log Complete

Figure 3-23. Save Complete Message

This changes the symbol on the log's folder from N to I, Figure 3-24. This indicates the log has been saved, and the status is Inactive.



TC100:VALUE
Description -
Data Source - TC100:VALUE
$HSTC100:VALUE-1-o

Figure 3-24. Save Completed Indication

This completes the configuration of the primary log.

Additional examples of primary log configurations are described in Section 3.2.8.5, Data Collection Examples:

- Example 1, Storing Instantaneous Values, No Compaction
- Example 2 - Storing Instantaneous Values Using Compaction
- Example 3 - Storing Calculated Values in a Primary Log

If you want to start (activate) the log, see Section 3.2.7, Starting/Stopping Logs.

3.2.4.2 Secondary Logs for Different Resolution/Time Span Views

A secondary log gets its data either from a primary log, or another secondary log. To add a secondary log, select the primary or secondary log that will be the data source for the new log you are adding, then right-click and choose **Add Numeric Log** from the context menu.

One application for secondary logs is to provide different views of a variable by configuring a different time span for the respective primary and secondary logs. For example, in Figure 3-25 the primary log stores high resolution data for a short time span, while the secondary logs store progressively lower resolution data for longer time spans. This configuration lets you store data over longer periods without using as much storage capacity.

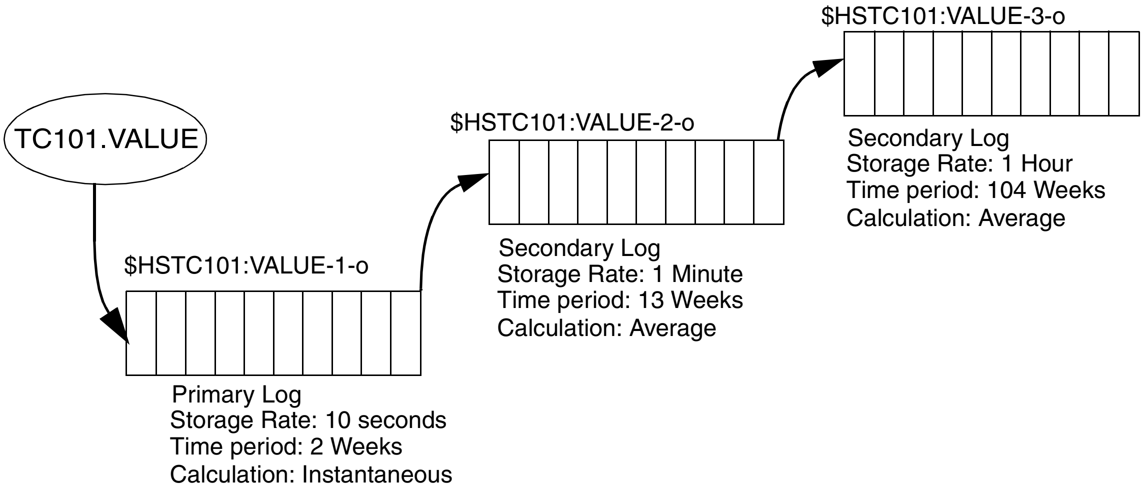


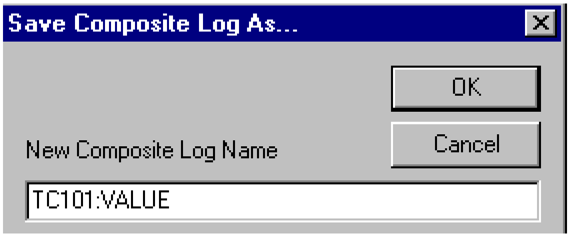
Figure 3-25. Example, Secondary Logs with Different Views of a Process Variable

To build a composite log like the one shown in Figure 3-25, start by adding a composite log, and then add a basic primary log.

You can start with a new composite log; however, if one with the required structure and log attribute definitions already exists, you can simply copy that one. This procedure is described below.

Creating a Composite Log by Copying an Existing One

Select the log that you want to copy from the Available Composite Logs List, then choose **File>Save Composite Log As** from the main menu bar. This displays the Save As dialog. Enter a unique Composite Log Name, Figure 3-26, then click **OK**.



New Composite Log Name: TC101:VALUE

Figure 3-26. Save Composite Log As Dialog

NOTE

The Available Composite Logs List will appear has if the new log as replaced the original log, but this is not the case. To see both logs, refresh the list by choosing **Get Full Log List** from the Available Composite Logs List right click menu.

Adding Secondary Logs

This composite log requires two secondary logs - one connected to the primary log, and the other connected to the first secondary log, Figure 3-27. To do this:

1. Right click on the primary log and choose **Add Numeric Log** from the context menu, Figure 3-27.

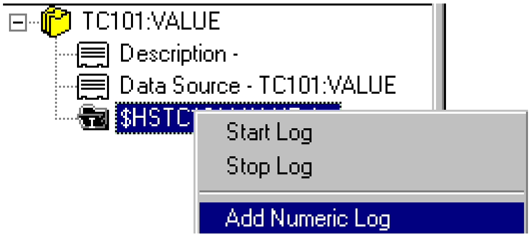


Figure 3-27. Adding a Secondary Log Under a Primary Log

This specifies the primary log as the data source for the first secondary log. This is indicated by the secondary log being slightly indented from the primary log, Figure 3-28.

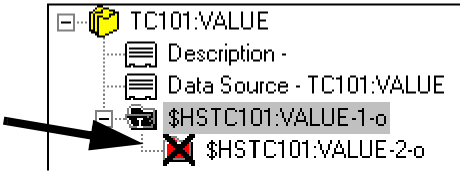


Figure 3-28. Secondary Log Indented

2. Add the other secondary log directly under the first secondary log. To do this, right click on the secondary log and choose **Add Numeric Log** from the context menu, Figure 3-29.

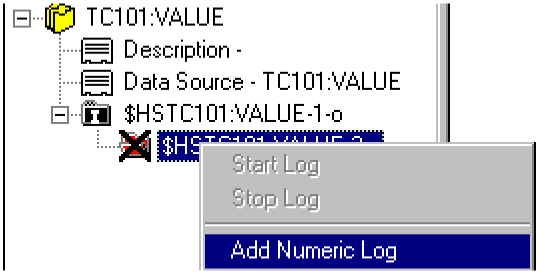


Figure 3-29. Adding a Secondary Log Under Another Secondary Log

This specifies the first secondary log as the data source for the second. When you are finished, the composite log structure will be as shown in Figure 3-30. Again, notice the second secondary log is slightly indented from the first.

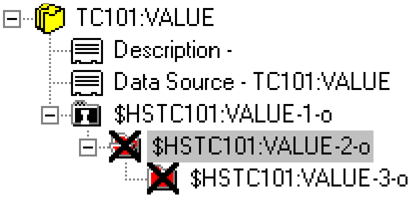


Figure 3-30. Secondary Logs Added Under Primary Log

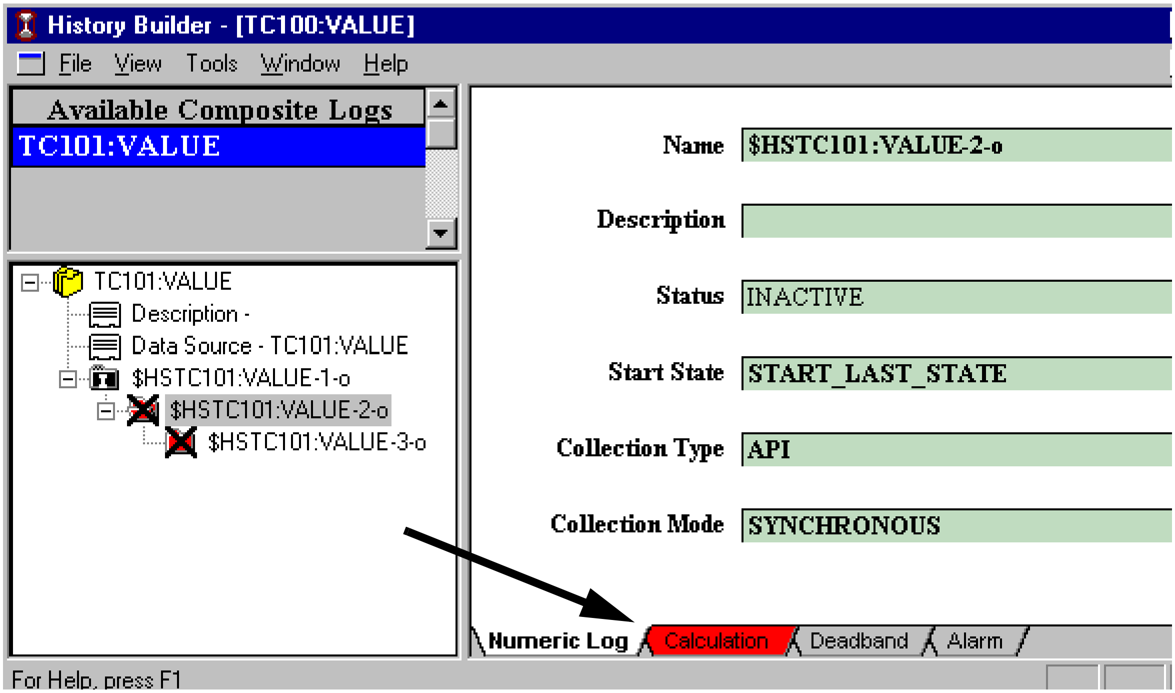
Configuring Log Attributes via Tabs

The primary log configuration is already complete since it was created by copying an existing log. However, both secondary logs are incomplete as indicated by the red folders. Again, you can configure the log attributes either via the Log Configuration wizard, by double-clicking an attribute in the Tree View, or via the log attribute tabs.

In this case it is easier to use the tabs. This is because, this application requires the Calculation Algorithm to be configured. The Log Configuration Wizard does not list this attribute since it is not mandatory that this attribute be configured. To configure the first secondary log using tabs:

1. Click on the first secondary log to make the log attributes accessible via tabs.

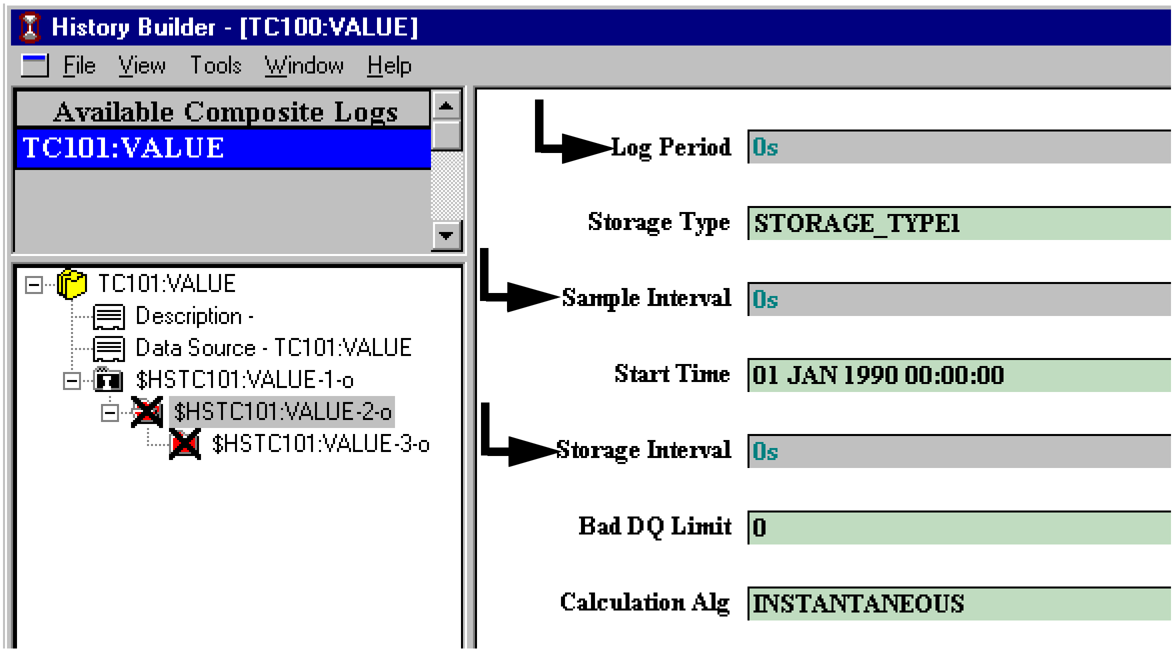
   Any tab that has unconfigured attributes is indicated in red. In this case, only the **Calculation** tab has attributes that must be configured, Figure 3-31.



Name: $HSTC101:VALUE-2-o
Description:
Status: INACTIVE
Start State: START_LAST_STATE
Collection Type: API
Collection Mode: SYNCHRONOUS

Figure 3-31. Red Tab Indicating Incomplete Configuration

2. Click the **Calculation** tab. Any unconfigured attributes are shown in red, Figure 3-32.



Log Period: 0s
Storage Type: STORAGE_TYPE1
Sample Interval: 0s
Start Time: 01 JAN 1990 00:00:00
Storage Interval: 0s
Bad DQ Limit: 0
Calculation Alg: INSTANTANEOUS

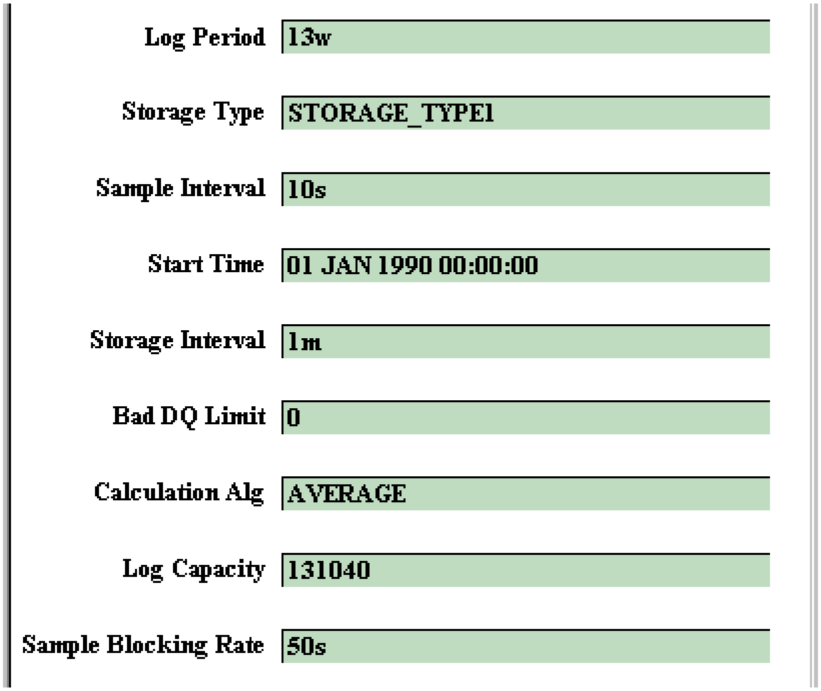
Figure 3-32. Unconfigured Attributes Indicated in Red

3. To configure attributes on a tab, simply click inside the corresponding field and enter the valid value. Configure the following attributes in the order indicated:
   - Log Period: **13w**
   - Calculation Algorithm: **Average**
   - Sample Interval: **10s**
   - Storage Interval: **1m**

NOTE

The order is not critical; however, you should always configure the Calculation Algorithm before the Sample Interval and Storage Interval. How the Calculation Algorithm is configured determines how you need to configured the other two attributes. See Section 3.2.8.2, Calculation Attributes for further details.

The completed configuration is shown in Figure 3-33.

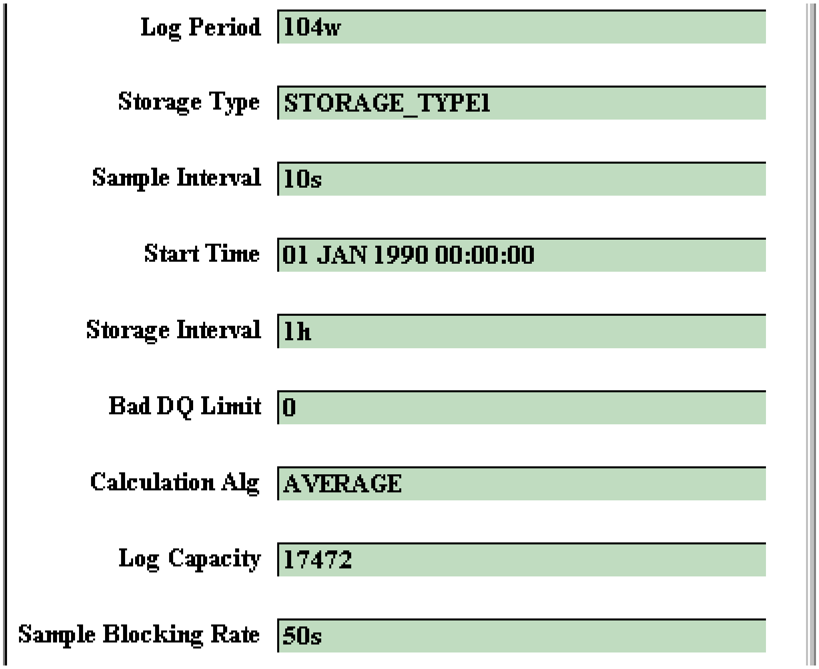


Log Period 13w
Storage Type STORAGE_TYPE1
Sample Interval 10s
Start Time 01 JAN 1990 00:00:00
Storage Interval 1m
Bad DQ Limit 0
Calculation Alg AVERAGE
Log Capacity 131040
Sample Blocking Rate 50s

Figure 3-33. Configuration of First Secondary Log Completed

4. Repeat steps 1 & 2 for the other secondary log:
   - Log Period: **104w**
   - Calculation Algorithm: **Average**
   - Sample Interval: **10s**
   - Storage Interval: **1h**

The completed configuration is shown in Figure 3-34.



Log Period 104w

Storage Type STORAGE_TYPE1

Sample Interval 10s

Start Time 01 JAN 1990 00:00:00

Storage Interval 1h

Bad DQ Limit 0

Calculation Alg AVERAGE

Log Capacity 17472

Sample Blocking Rate 50s

Figure 3-34. Configuration of other Secondary Log Completed

You can retain the defaults for the remaining attributes, or you can configure additional attributes to meet the requirements of your application. If you want to configure additional attributes, refer to Section 3.2.8.1, Numeric Log Attributes for details.

Remember to save your work before leaving this composite log. To save your work choose **File>Save Composite Log** from the main menu bar.

3.2.4.3 Primary and Secondary Logs with Calculations

Another application for secondary logs is to store multiple aspects of the logged variable such as maximum and minimum values for a specified time interval. This is done by applying different calculation algorithms to the respective primary and secondary logs.

The calculation is performed before the data are stored in the log. This allows you to store a single value that is representative of a larger time span of values, or key characteristics of the data. For example, you can sample a variable every 10 seconds and put those values into a primary log. Then you can configure that a 1-minute minimum, and maximum be calculated from data in the primary log and store those values in secondary logs.

This example is illustrated in Figure 3-35. In this case, both secondary logs use the primary log as their data source.

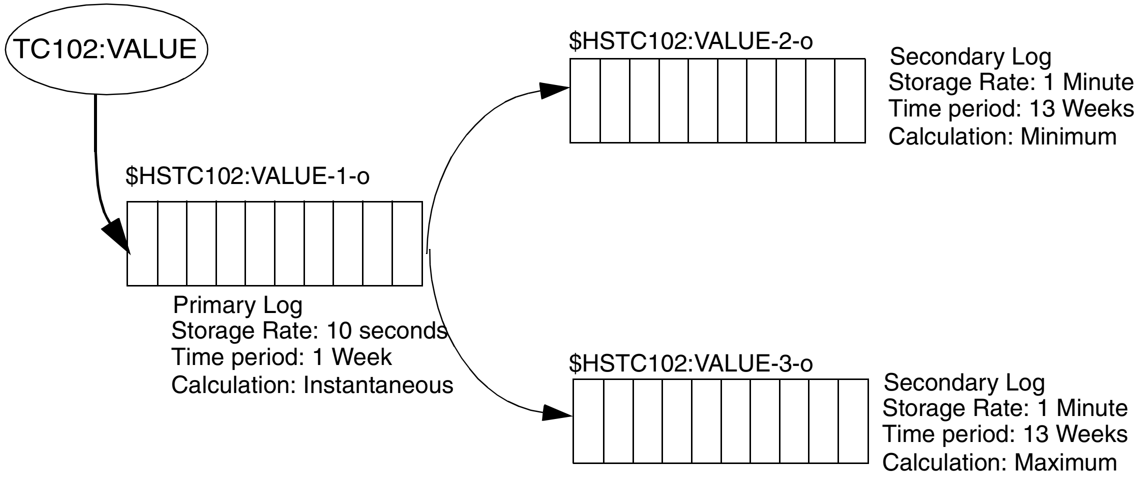


Figure 3-35. Example, Secondary Logs with Different Views of a Process Variable

Creating the Composite Log

Start by creating a new composite log and then adding a simple numeric log as described in Section 3.2.4.1, Simple Numeric Logs for Synchronous Process Data.

Adding Secondary Logs

This composite log structure also requires two one secondary logs; however, in this case, both are connected to the primary log, Figure 3-35. To do this:

1. Right click on the primary log and choose **Add Numeric Log** from the context menu, Figure 3-36.

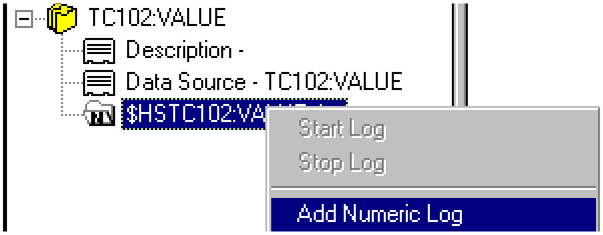


Figure 3-36. Adding a Secondary Log

2. Repeat step 1 to add the other secondary log, Figure 3-37.

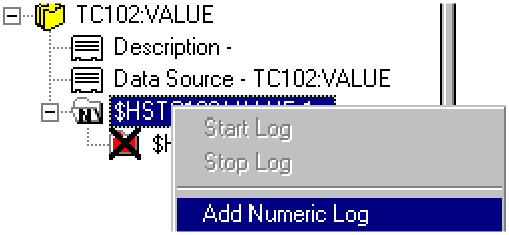


Figure 3-37. Adding Other Secondary Log

The finished composite log is shown in Figure 3-38. The secondary logs are flush rather than indented since both use the primary log as their data source.

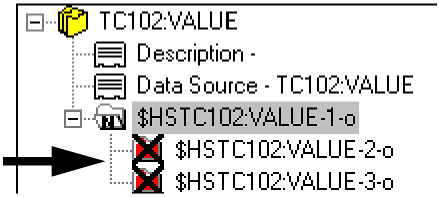
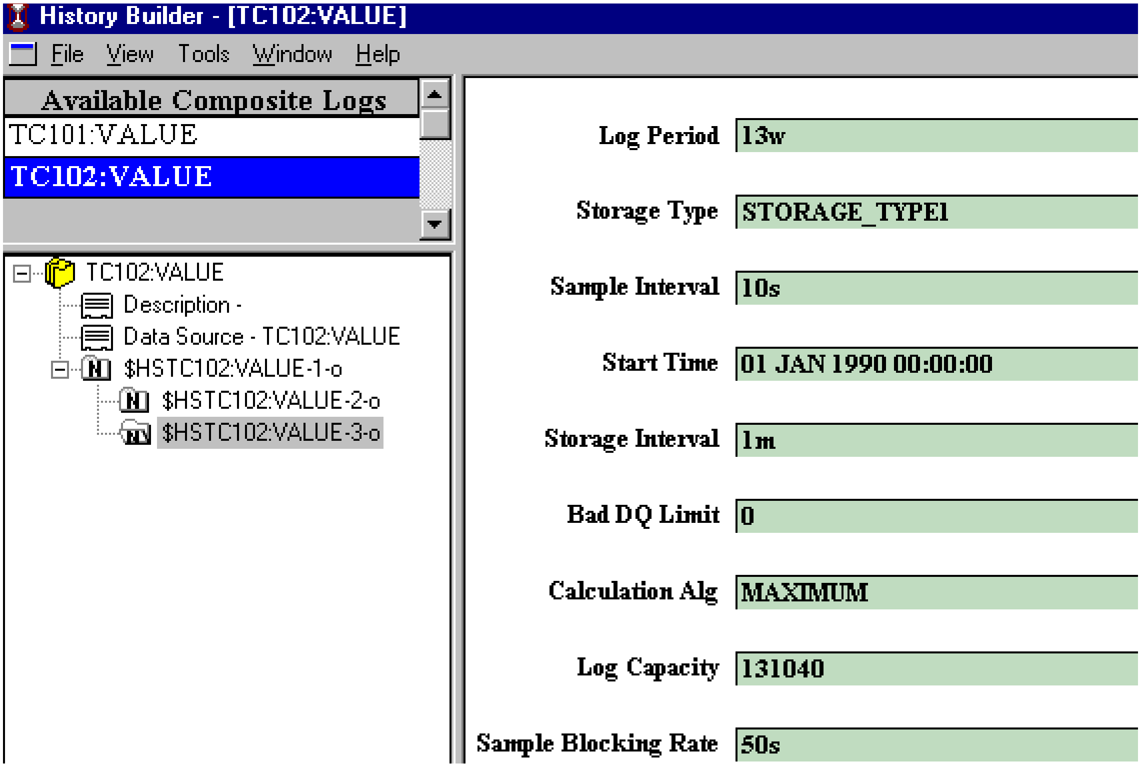


Figure 3-38. Two Secondary Logs Sharing Primary Log as Data Source

3. Using the tab method, configure the secondary logs as follows:
   - Log Period: **13w**
   - Calculation Algorithm: **Minimum** for one, **Maximum** for the other
   - Sample Interval: **10s**
   - Storage Interval: **1m**

The completed composite log configuration is shown in Figure 3-39.



Log Period	13w
Storage Type	STORAGE_TYPE1
Sample Interval	10s
Start Time	01 JAN 1990 00:00:00
Storage Interval	1m
Bad DQ Limit	0
Calculation Alg	MAXIMUM
Log Capacity	131040
Sample Blocking Rate	50s

Figure 3-39. Example, Composite Log using Calculations

You can retain the defaults for the remaining attributes, or configure additional attributes to meet the requirements of your application. If you want to configure additional attributes, refer to Section 3.2.8.1, Numeric Log Attributes for details.

Remember to save your work before leaving this composite log. To save your work choose **File>Save Composite Log** from the main menu bar.

Additional examples of primary/secondary log configurations are described in Section 3.2.8.5, Data Collection Examples:

- Example 4 - Storing Calculated Values in Primary and Secondary Logs
- Example 5- Storing Calculated Values with Compaction

3.2.5 Deleting Logs

To delete a composite log, select the log that you want to delete from the Available Composite Logs List, and then choose **File>DeleteComposite Log** from the main menu bar.

3.2.6 Adjusting Numeric Log Attributes

Individual log attributes can be configured using any of the following three methods:

- Configuring Log Attributes with the Log Configuration Wizard
- Configuring Log Attributes via Tabs
- Configuring Attributes from the Tree View

If you are having difficulty locating errors in an invalid log configuration, you can use the Log Configuration wizard. An example is provided in Configuring Log Attributes with the Log Configuration Wizard in Section 3.2.4.1, Simple Numeric Logs for Synchronous Process Data.

For details regarding a specific attribute see:

- Section 3.2.8.1, Numeric Log Attributes
- Section 3.2.8.2, Calculation Attributes
- Section 3.2.8.3, Deadband Attributes
- Section 3.2.8.4, Alarm Attributes

Limitations

When you configure a log, before you save the log you can edit most log attributes. You can not edit: Status, Log Capacity (for synchronous logs).

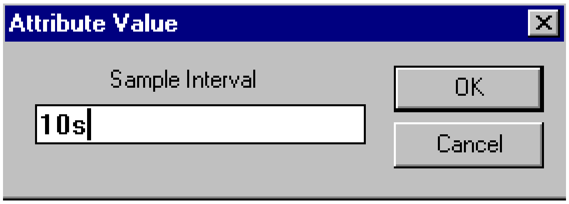
After you save the log, the following attributes can not be edited (in addition to the ones above): Composite Log Name, Numeric Log Name, Sample Interval, Storage Type.

It is recommended that you do not edit the presentation, engineering units, and alarm attributes as described in Section 3.2.8.4, Alarm Attributes.

Check that the log state is INACTIVE. If it is ACTIVE, deactivate the log. You are not allowed to change attributes in an ACTIVE log.

3.2.6.1 Configuring Attributes from the Tree View

Any log attributes that are selected to be shown in the Tree View can be configured directly from the Tree View. To do this, double-click on the attribute name under the folder for the individual log. This displays the Attribute Value dialog, Figure 3-40.



Sample Interval: 10s

Figure 3-40. Attribute Value Dialog

Select the appropriate value from the pull-down menu, or enter the value directly in the field, and then click **OK**.

NOTE

You can select which log attributes to show in the Tree View via the Tree Options command in the right-click menu. For details, see Section 3.2.2.1, Select Which Attributes to Show for Primary/Secondary Logs.

3.2.7 Starting/Stopping Logs

When you attempt to start a log, the log may be started immediately, or it may be delayed, depending upon how the log's Start Time attribute is defined. Start Time determines the earliest allowable time that the log can be started. In addition, the start time also determines the hour, minute, and second when the first point is collected. The status of the start request is indicated by color in the Tree View.

Starting/Stopping Individual Numeric Logs

You start and stop logs on an individual basis via the Tree View right-click menu. In the Tree View, right-click on the log that you want to start or stop, and then choose either **Start Log** or **Stop Log** from the context menu.

Starting/Stopping a Composite Log

To start or stop all the logs in a composite log at once, select the composite log and then choose **Start Log** or **Stop Log**.

3.2.8 Log Attributes

Log attributes are grouped under the following four categories:

- Section 3.2.8.1, Numeric Log Attributes
- Section 3.2.8.2, Calculation Attributes
- Section 3.2.8.3, Deadband Attributes
- Section 3.2.8.4, Alarm Attributes

Also refer to Section 3.2.8.5, Data Collection Examples. These examples provide guidelines and recommendations for common data collection/storage applications:

- Example 1, Storing Instantaneous Values, No Compaction
- Example 2 - Storing Instantaneous Values Using Compaction
- Example 3 - Storing Calculated Values in a Primary Log
- Example 4 - Storing Calculated Values in Primary and Secondary Logs
- Example 5- Storing Calculated Values with Compaction

3.2.8.1 Numeric Log Attributes

Numeric Log Name

The numeric log name is derived from the Composite Log Name that you specify when you create a new composite log. As you add primary and secondary logs, History assigns each one a unique numeric log name by adding a prefix `$HS` and a suffix `-`*n*`-o` to the Composite Log Name.

n is an integer that uniquely identifies different logs in a composite log. The number is assigned by History according to the order logs are added to the composite log. For example, a primary log with Composite Log Name TC100,MEASURE, has the log name $HSTC100,MEASURE-1-o. The next (secondary) log will have the log name $HSTC100,MEASURE-2-o, and so on.

The last character in the log name indicates whether the log is original (from disk) or restored from tape: o = original, r = restored

Description

This is an optional text string for documentation purposes only.

Status

This is a read-only field that indicates the log's status. Log status can be:

- **INACTIVE** - not storing data (default)
- ACTIVE - storing data
- TRANSLATING - log is trying to find the data source
- PENDING - log is activated, has found its data source (TRANSLATING = completed), and is waiting to start collecting data based on Alignment Time.

Individual logs can be activated or de-activated via the Tree View right click menu. See Section 3.2.7, Starting/Stopping Logs.

Secondary logs go to active immediately; however, they do not start collecting and storing data until the primary log becomes active.

Start State

This is the initial state for the log when the node is started or restarted. The choices are:

- START_INACTIVE
- START_ACTIVE
- **START_LAST_STATE** (default) - restart in state log had when node shut down

Collection Mode

The collection mode determines how data is processed when it is received by History. This field is not configurable. The collection mode is based on the type of log that was added. The only applicable mode for this release is *synchronous*.

SYNCHRONOUS - This is periodic *Time Sequence* data. This is the collection mode for real-time process variables whose data is collected via numeric logs. The time interval between samples received by History must always be the same. Calculations and deadband can be applied to synchronously collected data.

Collection Type

This determines the application that History collects data from. Collection type is only configurable for primary logs. The options are:

- OPC - History periodically collects data from an OPC data server. A number of pre-determined OPC servers are supported. These are defined in a configuration file. To view and/or modify this configuration file, see Section 5.3, Configuring OPC Server Access.

  Typically, data collection is via the Advant OPC server.
- API - History periodically collects data using OMF via the Real-time Accelerator Board.

3.2.8.2 Calculation Attributes

The data collection function is illustrated in Figure 3-41. This function collects values and buffers them internally for a period of time according to the sample blocking rate. It then performs the specified calculation on the buffered data. A value is calculated for every storage interval. These calculated values are passed to the deadband function which checks to see if the value is within or outside the deadband range. If deadband is not active, the value is sent to storage. If deadband is active, the value is sent to storage if any one of the following are true:

- The value is outside the deadband range.
- The value is within the deadband range, but the deadband storage interval time has elapsed since the last value was stored.
- The value status changes (possible statuses are: no data, bad data, good data)

If none of these criteria are met, the value is not stored.

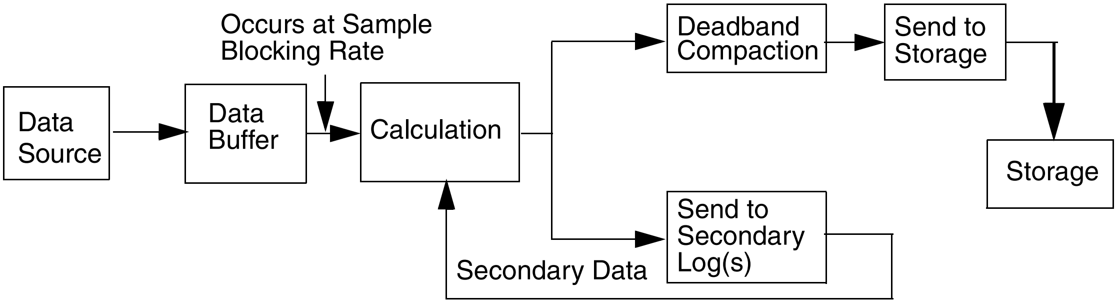


Figure 3-41. Data Collection, Functional Diagram

Figure 3-41 shows that data sent to a secondary log does not undergo compaction, even if the primary log has deadband configured. The secondary log gets all values that result from the calculation. Thus, the calculation performed by the secondary log is more accurate than if done using just data stored after deadband compaction.

You can configure History to perform calculations on collected data before the data are stored in the log. When a calculation is active, the specified algorithm, such as summing or averaging, is applied to a configured number of data samples from the data source. The result of the calculation is stored by History.

For example, you can configure History to collect from a data source at a specified sample interval and storage interval, calculate the average value over the storage interval, and then store the calculated average value.

You can also configure History for data compaction to reduce the amount of data stored for numeric variables. Both calculations and deadband can be used concurrently for a log. The calculation is done first. Then the deadband algorithm is applied to see if the result should be stored.

For examples of data collection applications, see Section 3.2.8.5, Data Collection Examples.

Log Period

Disk space is allocated to logs according to their log capacity, which is a function of the log time period and storage interval. The log period can be adjusted on an individual log basis to keep log capacity at a reasonable size for each log. For all synchronous logs with or without data compaction, log capacity is calculated as: log capacity = (log period/storage interval).

When you use compaction, the log period is effectively increased so that you store the same number of samples (as determined by the log capacity) over a longer period of time.

Enter the log period as an integer value ≤ 4095 with time unit indicator: w or W= week, d or D= day, h or H= hour, m or Minutia, s or S= second. The maximum is 4095w.

Storage Type

This is the method for storing log entries. The choices are:

- TYPE1 - File-based storage, 24 bytes per entry. Full status information for each entry.
- TYPE2 - File-based storage, 4 bytes per entry. Abbreviated status information for each entry.

For further details, see Section 3.2.9, File Storage.

Sample Interval

This is the rate at which data is sampled from a data source. Whether or not the sample is stored depends on whether or not a calculation algorithm is specified, and whether or not a compaction deadband is specified.

If data is not received within 50% of the interval prior to, or 100% of the interval after the expected time, the entry value is marked with status NO_DATA. See Section 3.2.8.2, Calculation Attributes.

Enter the sample interval as an Integer value with a lower or upper case time unit indicator: w (week), d (day), h (hour), m (minute), or s (second). For example: **5s** or **5S**, **15m** or **15M**. Monthly is NOT a legal sample interval.

If the time stamp of a received value is greater than or equal to 50% before, or greater than or equal to 100% after the sample interval, the status of the stored data is NO_DATA. Time stamps may drift due to load in the system. A case where time drift can cause NO_DATA entries to be stored is illustrated in Figure 3-42.

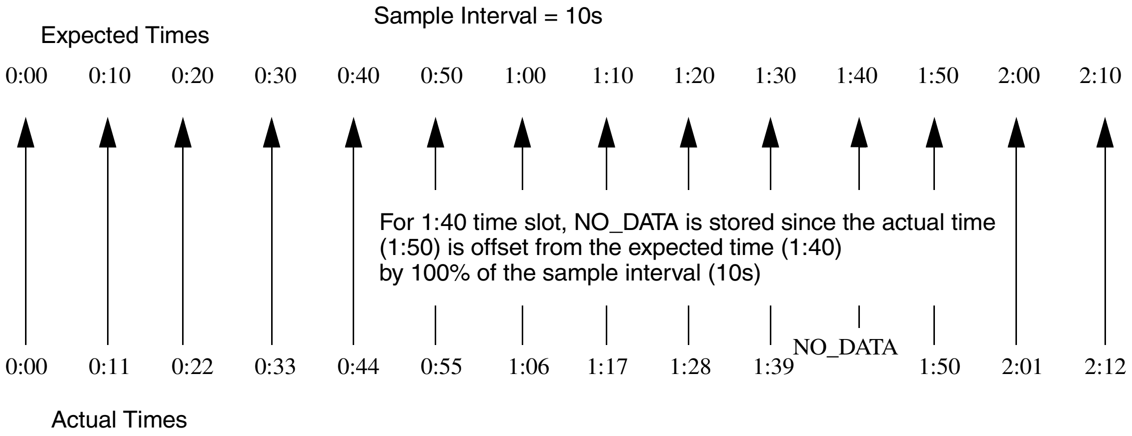


Figure 3-42. Example of Time Drift

Start Time

This is the earliest allowable system time that the log can become active. If the specified start time has already past, the log will begin to store data at the next storage interval. If the start time is in the future, the log will go to PENDING state until the start time arrives. Enter the start time in the following format:

JAN-01-90 00:00:00 (default)

NOTE

The order of the month, day, and year and the abbreviation of the month is standardized based upon the language you are using.

You can use the start time to distribute the load. Both the blocking rate and start time may be defined initially when you configure the History database. If you do not configure these attributes initially, or if you find that adjustments are required, the Stagger Collection function in the hsDBMaint utility provides a convenient means for adjusting these log attributes on-the-fly. Refer to Section 5.2.10, Stagger Collection and Storage.

It is recommended NOT to use small differences (less than one minute) to distribute the load. Managing log collection at slightly different times will result in significant overhead.

The time when a log is scheduled to start collecting data is referred to as the log's *alignment time*. Alignment time is the intersection of the log's start time and its storage interval. For instance if the start time is Jun-30-95 10:00:00, and the storage interval is15 s, the log will store data at 00, 15, 30, and 45 seconds of each minute starting at 10:00:00 on June 30, 1995. In this case, this is the alignment time.

If the start time for a log has already passed, it will be aligned based on the storage interval. For example, a log with a storage interval of 1m will store data at the beginning of each minute. So its alignment time is the beginning of the next minute.

The log stays in PENDING state until its alignment time. For the log whose storage interval is 1 minute, it will be PENDING until the start of the next minute.

The default is: **JAN-01-90 00:00:00**. The order of the month, day, and year and the abbreviation of the month is standardized based upon the language you are using.

Storage Interval

This is the time interval at which data is stored in the log, except when deadband is active. Enter the storage interval as an Integer value with a lower or upper case time unit indicator: w (week), d (day), h (hour), m (minute), or s (second). For example:**10s** or **10S**. The maximum is 4095w. The storage interval must be a multiple of the sample interval. For example, if the sample interval is 5s (five seconds), the storage interval can be 10s, 15s, 20s, 25s, and so on. A special storage interval is monthly. This causes a value to be stored at the end of the month.

The timestamp of a monthly log depends upon the storage interval and alignment of its data source.

If the data source is a daily log which is aligned to 06:00, then the monthly log values will have a time of 06:00.

If the data source is hourly or less, then the monthly log will be around midnight (depending on the alignment of the data source). For example, an hourly log at 30m after the hour will produce a monthly value at 00:30.

If the data source is an 'x'h log aligned to 06:00 (for example, 8h log with values stored at 06:00, 14:00, 22:00, and so on), then the monthly log will have a time of 06:00.

Bad Data Quality Limit

When the percentage of samples with bad data quality or no data status used in a calculation exceeds the Bad Data Quality Limit, the data quality of the calculated result is bad. For example, 20% means that 2 out of 10 data samples can be bad.

This attribute can be used to avoid having data stored with BAD_DATA quality. When sampling remote objects, the interval between values may vary depending upon load in source node. In this case the sampled data will have status BAD_DATA. Use of this attribute allows some of the sampled values to be bad and still get a stored value with ok status.

For example, set sample interval to 3s storage interval to 30s and make average. Set Bad Data Quality to 30%. In this case, at least three of the ten values must have bad status in order for the calculation result to be bad.

Enter the bad data quality limit as a percent. For example **10.500** is 10.5%. The default is 0.

Calculation Algorithm

The following types of calculations performed on the data before it is stored:

AVERAGE
MAXIMUM
MINIMUM
INSTANTANEOUS - no calculation (default)
SUM
SUM_OF_ SQRS
STANDARD DEVIATION
NUM_OF_ VALUES

You can specify calculations to be done before data is stored in a log (either primary or secondary), Figure 3-43. This lets you store a single value that is representative of a larger time span of values, or key characteristics of the data. For example, you can sample a variable every minute and put those values into a primary log. Then you can configure that an hourly average, minimum, and maximum be calculated from data in the primary log and store those values in secondary logs.

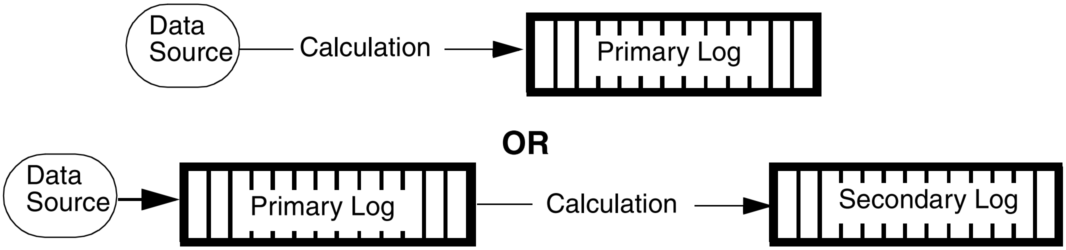


Figure 3-43. Calculations

If a calculation algorithm other than INSTANTANEOUS is specified for the log, the algorithm is performed at each storage interval. The storage interval must be a multiple of the sample interval. For example, if the sample interval is 5s (five seconds), the storage interval can be 10s, 15s, 20s, 25s, and so on.

When the calculation is performed, the specified number of data samples are entered in the algorithm, and the result is passed to the deadband function. If the calculation algorithm is INSTANTANEOUS (no calculation), the sample and storage intervals must be equal, and each data sample is treated as a result.

Log Capacity

This is the maximum number of log entries for a log. If a log is full, new entries replace the oldest entries.

For synchronous collection, this field is not configurable. Log capacity is calculated as follows. Log Capacity = (Log Period/ Storage Interval). See Section 3.2.8.2, Calculation Attributes for information on Log Period.

Sample Blocking Rate

This is the amount of time that data is buffered in collection before it is sent to storage. The sample blocking rate is only used in primary logs.

Enter the Sample Blocking rate as an Integer value with a lower or upper case time unit indicator: **w** (week), **d** (day), **h** (hour), **m** (minute), or **s** (second). The Sample Blocking Rate must be a multiple of the storage interval. For example: **10h** or **10H**. The maximum is **4095w**.

Sample Blocking Rate is used in different ways, depending upon the type of data source. Most synchronous primary logs use blocking rate to allocate a buffer in the collection function to hold the blocking rate worth of samples before applying a calculation and (if applicable) deadband algorithms and sending the samples to storage. Each blocked sample consumes 28 bytes of memory.

The blocking rate can be used to phase data from logs with similar sample intervals by using prime number multiples of the sample interval. For example, if a system has 1000 logs, and all logs have a 10 minute blocking rate, every 10 minutes storage would be flooded with 1000 messages. This may degrade performance. Instead you can use blocking rate to distribute the load over two or more intervals. For instance you can assign 500 logs a 9-minute blocking rate, and the other 500 logs an 11-minute blocking rate. This way, the flood of 1000 messages would be limited to once every 99 minutes (9x11), rather than once every 10 minutes.

Sample blocking rate is NOT used under the following conditions:

- if the storage interval is Monthly
- if the Collection Type is USER_SUPPLIED. In this case the values are stored as they are sent to History.

3.2.8.3 Deadband Attributes

Deadband compaction has two advantages:

- It reduces the load on the system by filtering out values that do not have to be stored since there is not a significant change in the value.
- It allows data to be stored on-line for a longer period of time since the data is not replaced (overwritten) as quickly.

Compaction is specified on an individual log basis. You may use it for some logs and not for others. Compaction should not be used indiscriminately in all cases.

- If you want an accurate representation of your process over time, and you do not require that every data sample be stored, you can use compaction.
- If your application requires that every data sample be stored as an instance in the log, you cannot use compaction.

Deadband can be used on digital data sources so that only state changes are saved. The Deadband filters out insignificant changes in the process variable.The deadband defines the minimum change in value or in rate of change for the change to be significant. The deadband is defined as a percent of the engineering units range.

For example, if a process variable is scaled from 200 to 500, the range is 300. If the deadband is 2%, the minimum significant change is 6 (2% of 300). With these values, if the last stored sample was 400, and the next sample is 397, then 397 is within the deadband and will not be stored. If the next sample is 406.2, it will be stored. A small deadband results in more stored samples than a large deadband.

Deadband is activated by configuring a compaction deadband greater than zero. When deadband is active, the deadband algorithm determines whether the calculation result, or raw value if calculation algorithm is instantaneous, is within or outside the deadband. If the most recent result is outside the deadband, the previous result is stored. If the most recent result is within the deadband, it is held until the next result is calculated, and the previous result is discarded. If the compaction deadband is zero, so compaction not activated, all results are stored.

A deadband storage interval is defined on an individual basis for each log. The deadband storage interval is a multiple of the log's storage interval. It is the maximum time that can elapse without storing a sample when compaction is activated. Even if the process is so stable that there are no samples outside the deadband, samples will still be stored at the deadband storage interval. The deadband storage interval is always measured from the time the last sample was stored. The deadband storage interval should be at least 10 times larger than the storage interval to ensure a high compaction ratio.

When deadband is active, a sample may be stored for any of the following reasons:

- sample outside the deadband
- value status changed (possible statuses are: no data, bad data, good data)
- deadband storage interval elapsed since last stored sample

Figure 3-44 shows how samples are stored when deadband is active. The deadband function holds each sample until it determines whether or not the sample is required to plot the trend. Samples that are not required are discarded.

A sample is stored when deadband is activated at 0s. The sample at 10s is held until the next sample at 20s. Since there is no change in slope between 0s and 20s, the 10s sample is discarded and the 20s sample is held. This pattern is repeated at each 10-second interval until a change in slope is detected at 50s. When this occurs, the samples at 40s and 50s are stored to plot the change. At 90s another change in slope is detected, so the samples at 80s and 90s are stored.

Even though there are spikes between 80s and 90s, these values were not sampled so the slope remains constant (as indicated by the dotted line). Another change in slope is detected at 100s. The sample at 100s is stored to plot the slope. The sample at 90s, also used to plot the slope, was already stored. The slope remains constant thereafter. The next sample stored is 140s since the Deadband Storage Interval (40s) elapses between 100s and 140s. The sample at 180s is also stored due to the Deadband Storage Interval.

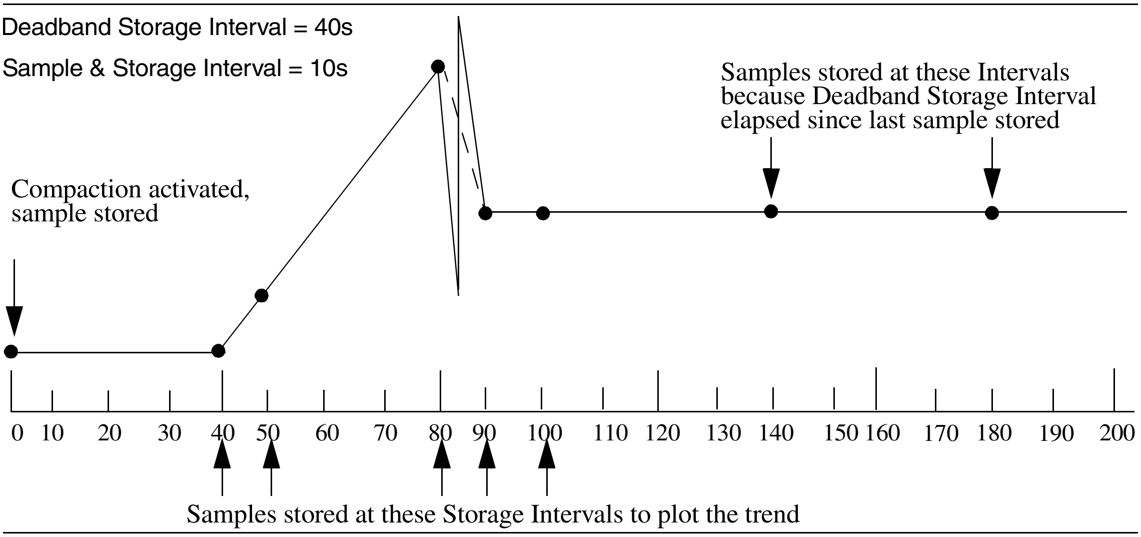


Figure 3-44. Data Compaction Example

Compaction effectively increases the log period so the same number of samples stored (as determined by the log capacity) cover a longer period of time. If your process is very unstable such that every sample is stored and there is no data compaction, the effective log period will be equal to the configured log period. If your process is perfectly stable (maximum data compaction, samples stored only at deadband storage interval), the effective log period is extended according to the following formula:

> effective log period = (deadband storage interval/storage interval) * configured log period

In most cases, process variation is not at either extreme, so the effective log period will fall somewhere in between the configured (minimum) and maximum value.

The calculated effective log period is stored in the EST_LOG_TIME_PER attribute. This attribute is updated periodically according to a schedule which repeats as often as every five days. If necessary, you can manually execute the update function via the hsDBMaint utility. This is described in Section 5.2.5, Update Deadband Ratio On-line.

When requesting by access name, EST_LOG_TIME_PER determines which log to retrieve data from (seamless retrieval). For instance, consider a case where the primary log stores data at 30 second intervals over a one-day time period, and the secondary log stores data at one-hour intervals over a one-week time period. The data compaction ratio is 5:1 which extends the effective log period of the primary log to five days. If you make a request to retrieve three-day-old data, the History retrieval function knows to get the data from the primary log rather than the secondary log, even through the configured log period of the primary log is only one day.

When EST_LOG_TIME_PER is updated, both internal History memory and the history database are updated with the new values of EST_LOG_TIME_PER and COMPACTION_RATIO. This will ensure the information will be available after a re-start of the History node.

Before using compaction, see examples 2 & 5 in Section 3.2.8.5, Data Collection Examples.

Compaction Ratio

This is a read-only attribute that indicates the number of entries sampled for each entry stored. It is filled in by running hsDBMaint. See Section 5.2.5, Update Deadband Ratio On-line.

Deadband Storage Interval

When deadband is active, this is the maximum time elapsed between storing entries. This assures that some data is stored, even if the process is stable.

Enter deadband storage interval as an Integer value with a lower or upper case time unit indicator: w (week), d (day), h (hour), m (minute), or s (second). It must be a multiple of the storage interval. For example: **1s** or **1S**.

The deadband storage interval defaults to ten times the storage interval. When the storage interval is less than or equal to 10m, the deadband storage interval should be much greater than ten times the storage interval. For example:

- if the storage interval = 5s, make the deadband storage interval at least 1h
- if the storage interval = 10m, make the deadband storage interval at least 8h

When the storage interval is greater than 10m, you can reduce the deadband storage interval to have less time elapse without storing a value.

Expected Ratio

This indicates the expected ratio for number of entries sampled and each entry stored.

Deadband Compaction

This is the minimum change between a sample and previously stored value that causes the new sample to be stored. Enter this as a percent of engineering units range (Range Max. - Range Min.). For example, 1 means 1% of range.

Estimated Period

This is a read-only attribute that indicates the time between the oldest and newest stored values. This is filled in by running hsDBMaint. See Section 5.2.5, Update Deadband Ratio On-line.

3.2.8.4 Alarm Attributes

The engineering units and alarm values are present in the log for information purposes only. The log does not scale variables or check for alarms. The attributes can be read by an application program that needs this information about the data source.

Presentation Maximum Value

This is the high value in setting the range for a trend grid. The value is in engineering units. Three decimal places are allowed, and the maximum allowed value is 99999999.

Presentation Minimum Value

This is the low value in setting the range for a trend grid. The value is in engineering units. Three decimal places are allowed, and the minimum allowed value is -9999999.

Engineering Units

This is the label for the value. It is a text string, 10 characters maximum.

State Set

This identifies the descriptor set for a device loop. It is the name of descriptor set.

Engineering Maximum Value

This is the high value of engineering range. The value is in engineering units. Up to three decimal places. Max. value is 99999999.

Engineering Minimum Value

This is the low value of engineering range. The value is in engineering units. Up to three decimal places. Max. value is 99999999.

High High Limit

This is the trip point for second high alarm. The value is in engineering units. Up to three decimal places. Max. value is 99999999.

High Limit

This is the trip point for first high alarm.The value is in engineering units. Up to three decimal places. Max. value is 99999999.

Low Limit

This is the trip point for first low alarm. The value is in engineering units. Up to three decimal places. Max. value is 99999999.

Low Low Limit

This is the trip point for second low alarm. The value is in engineering units. Up to three decimal places. Max. value is 99999999.

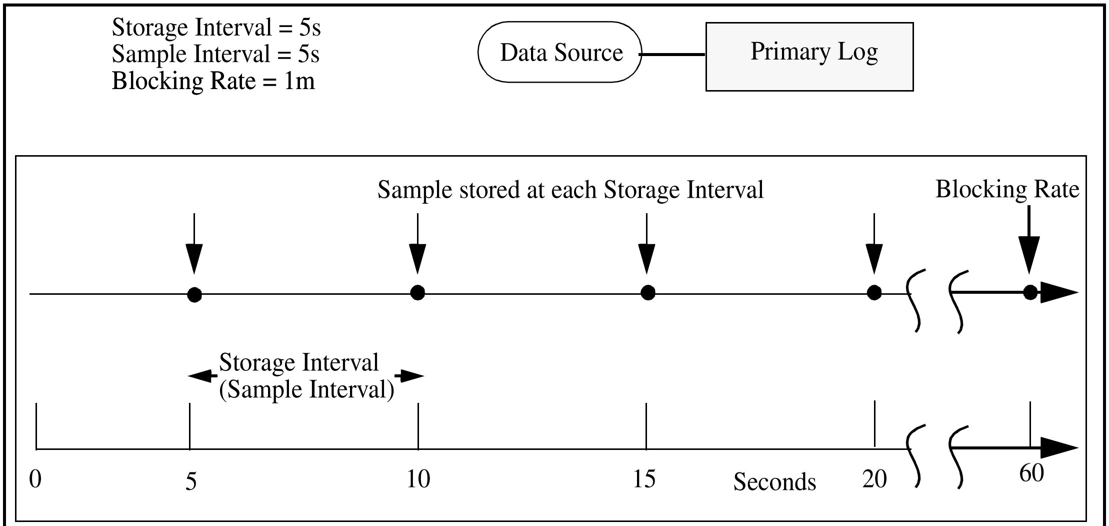
3.2.8.5 Data Collection Examples

The following data collection examples are described:

- Example 1, Storing Instantaneous Values, No Compaction
- Example 2 - Storing Instantaneous Values Using Compaction
- Example 3 - Storing Calculated Values in a Primary Log
- Example 4 - Storing Calculated Values in Primary and Secondary Logs
- Example 5- Storing Calculated Values with Compaction

Example 1, Storing Instantaneous Values, No Compaction

When the calculation algorithm is INSTANTANEOUS (no calculation), the sample and storage interval must be the same. If no compaction is specified, a data instance is stored every storage interval. Figure 3-45 shows the relation between these attributes of the Primary Log. In this case a sample is stored every five seconds. The blocking rate is 60 seconds. This means that the data samples to be stored are held in a buffer, and then written to disk in blocks of 12 samples at each 60-second interval. While the data is held in the buffer, it is not available for data retrieval.



In this case, each sample is saved for storage.
For the 60-second time period in this example,
12 sample values are written to disk.

Figure 3-45. Storing Instantaneous Values, No Compaction

Example 2 - Storing Instantaneous Values Using Compaction

In this example, the process variable is scaled from 200 to 500, the range is 300. The deadband is 2%, the minimum significant change is 6 (2% of 300). Again, since there is no calculation, the sample and storage interval must be the same. At each storage interval the data is checked by the deadband function. If a significant change in slope is detected, the sample is stored. Samples are discarded as the slope remains within the deadband. However, if the deadband storage interval (in this example, 25s) elapses since the last stored sample, another sample is stored on the deadband storage interval whether or not a change in slope is detected.

In Figure 3-46, the data is sampled every five seconds. Of the 17 samples taken, only 7 are stored. A summary is provided in Table 3-1.

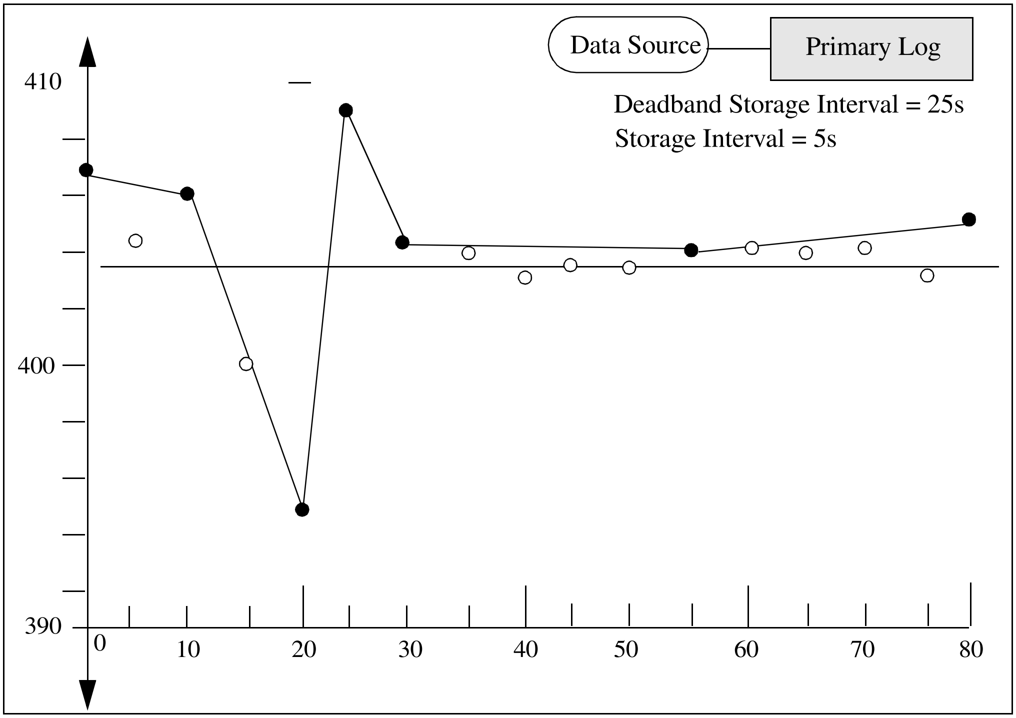


Figure 3-46. Storing Instantaneous Values w/Data Compaction

Table 3-1. Storing Instantaneous Values w/Compaction

Interval	Value	Stored	Reason
0	406.7	Yes	Deadband activated, first sample stored
5	404.5	No	No significant change in slope between 5s & 10s
10	406.2	Yes	Change in slope, next sample outside deadband
15	399.8.	No	No significant change in slope between 15s & 20s
20	394.9	Yes	Change in slope from 20 to 25, sample @ 25 outside deadband
25	409.0	Yes	Change in slope from 25 to 30, sample @ 30 outside deadband
30	404.2	Yes	Change in slope from 30 to 35, sample @ 35 outside deadband
35	404.0	No	No significant change in slope between 35s & 40s
40	402.8	No	No significant change in slope between 40s & 45s
45	403.3	No	No significant change in slope between 45s & 50s
50	403.2	No	No significant change in slope between 50s & 55s
55	404.1	Yes	Deadband storage interval (25s) elapsed since last stored sample (at 30s)
60	404.1	No	No significant change in slope between 55s & 60s
65	403.8	No	No significant change in slope between 60s & 65s
70	404.1	No	No significant change in slope between 605 & 70s
75	402.9	No	No significant change in slope between 70s & 75s
80	405.0	Yes	Deadband storage interval elapsed since last stored sample

Example 3 - Storing Calculated Values in a Primary Log

This example shows how to do a calculation before storing a value in a primary log, Figure 3-47. In this example, an average is calculated over an eight-hour shift.

The storage and sample intervals determine how many samples to use in the calculation (storage interval/sample interval). The storage interval must be an integer multiple of the sample interval. In this case, eight samples are used in each calculation.

Eight samples at hours 1,2,3,4,5,6,7, and 8 are averaged and the average is stored in the primary log at hour 8 (storage interval). Similarly, eight samples for two other sample groups are averaged and stored. The results of these three average calculations are written to disk at 24 hours (blocking rate). The process is repeated at 48 hours and averages are stored at 32, 40, and 48 hour.

.

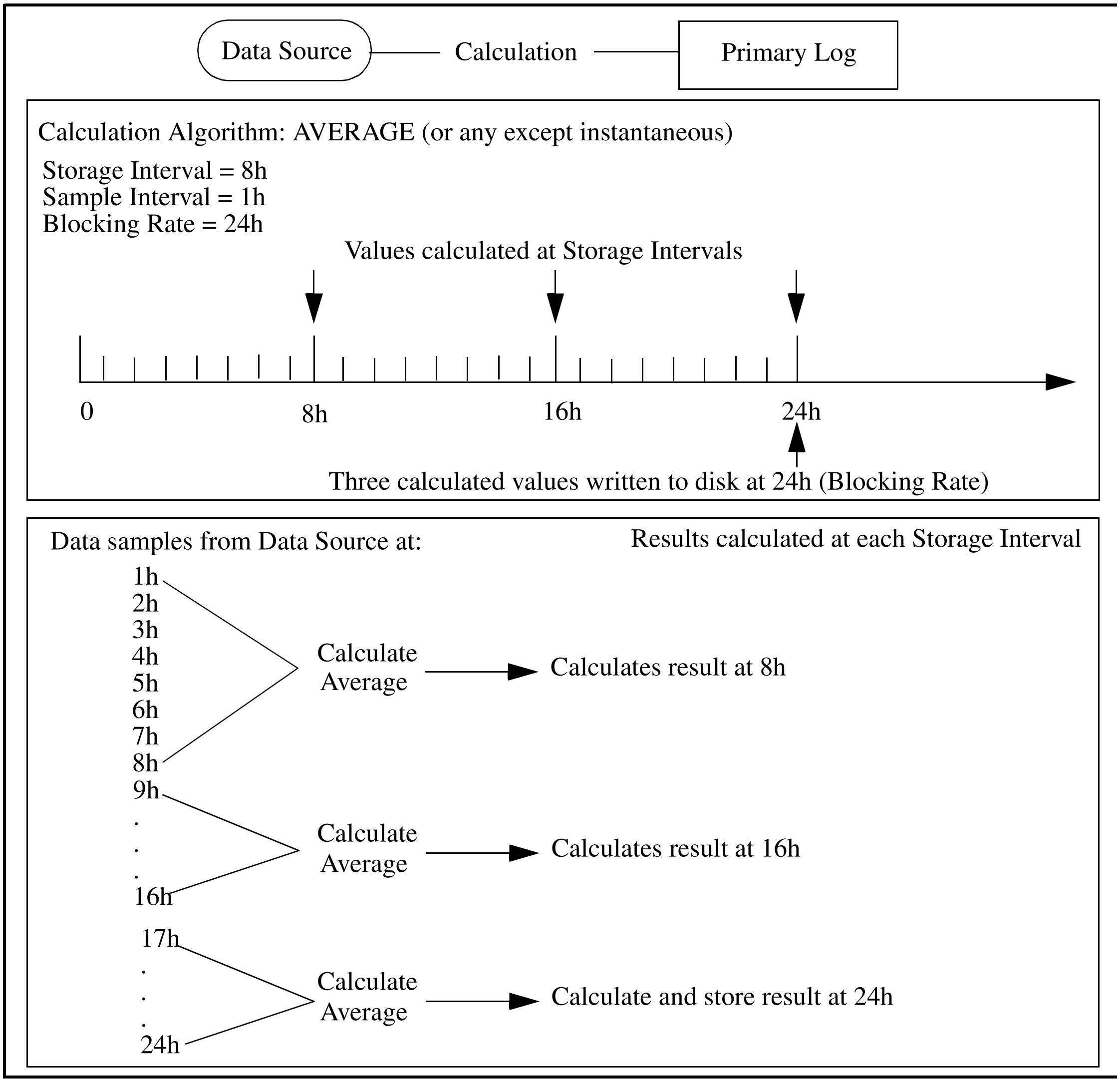

Data Source
Calculation
Primary Log
Calculation Algorithm: AVERAGE (or any except instantaneous)
Storage Interval = 8h
Sample Interval = 1h
Blocking Rate = 24h
Values calculated at Storage Intervals
0
8h
16h
24h
Three calculated values written to disk at 24h (Blocking Rate)
Data samples from Data Source at:
Results calculated at each Storage Interval
1h
2h
3h
4h
5h
6h
7h
8h
9h
16h
17h
24h
Calculate Average
Calculates result at 8h
Calculate Average
Calculates result at 16h
Calculate Average
Calculate and store result at 24h

Figure 3-47. Primary Log Example: Calculation being Used

Example 4 - Storing Calculated Values in Primary and Secondary Logs

This example shows how to do a calculation before storing a value in a secondary log. The primary log calculates the shift average as described in Example 3, and shown in Figure 3-47. The secondary log then calculates a daily average as illustrated in Figure 3-48.

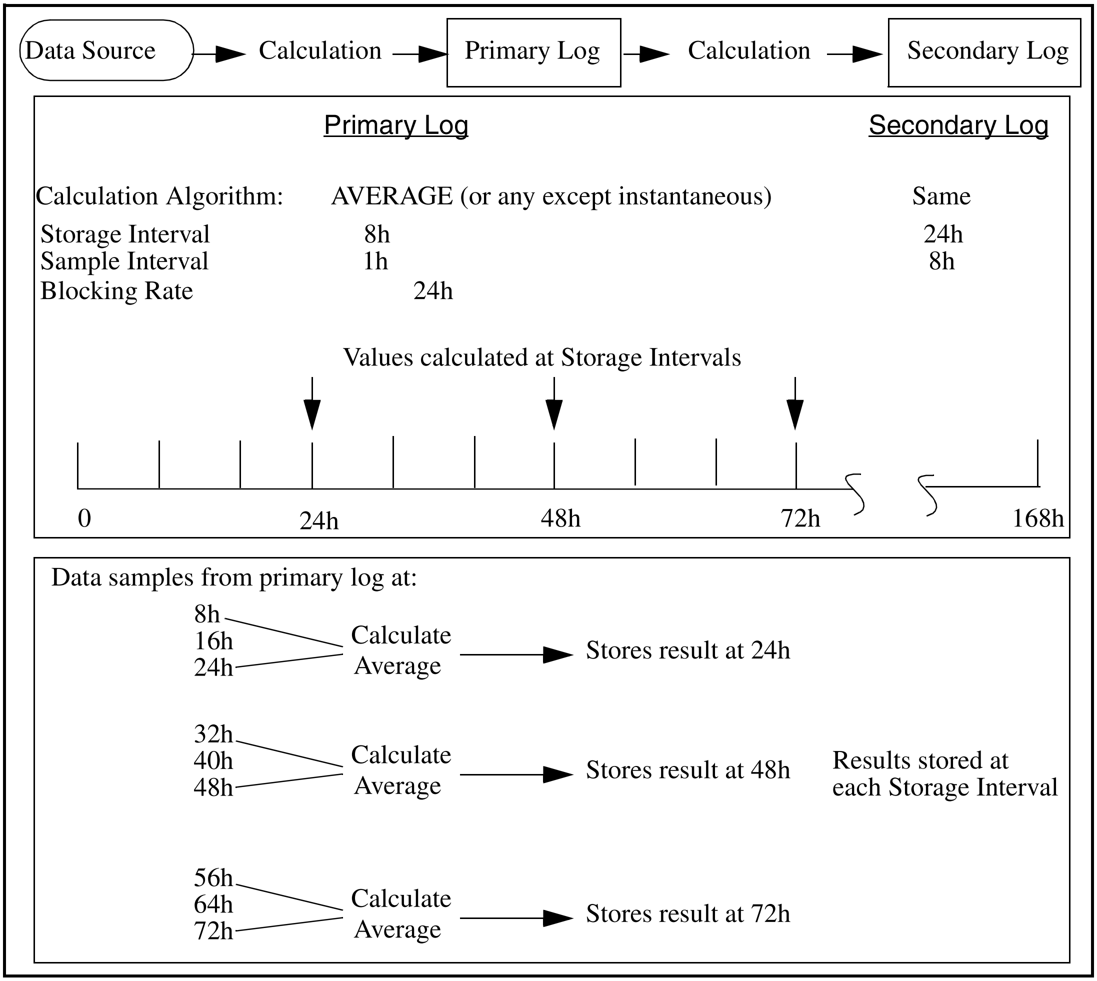


Figure 3-48. Secondary Log Example: Calculation being Used

NOTE

The sample blocking rate is not used for secondary logs. The sample blocking rate from the primary log determines when values are stored for all logs in the composite log.

Example 5- Storing Calculated Values with Compaction

This example uses the same data as Example 2 - Storing Instantaneous Values Using Compaction. The process variable is scaled from 200 to 500, the range is 300. The deadband is 2%, the minimum significant change is 6 (2% of 300). Since a calculation is being performed at each storage interval, the storage interval must be an integer multiple of the sample interval. In this case, data samples are collected at 5-second intervals, and four samples are used to calculate an average at 20-second intervals. At each storage interval the calculated value is checked by the deadband function. If a significant change in slope is detected, the calculated value is stored. Calculated values are discarded as the slope remains within the deadband. However, if the deadband storage interval (in this example, 60s) elapses since the last stored sample, another sample is stored at that time whether or not a change in slope is detected.

In Figure 3-49, calculations are performed at 20-second intervals. Of the 4 calculations performed, only 2 are stored. A summary is provided in Table 3-2.

Notice the contrast in the trends between Figure 3-46 and Figure 3-49, even though the same data is used to plot the trends.

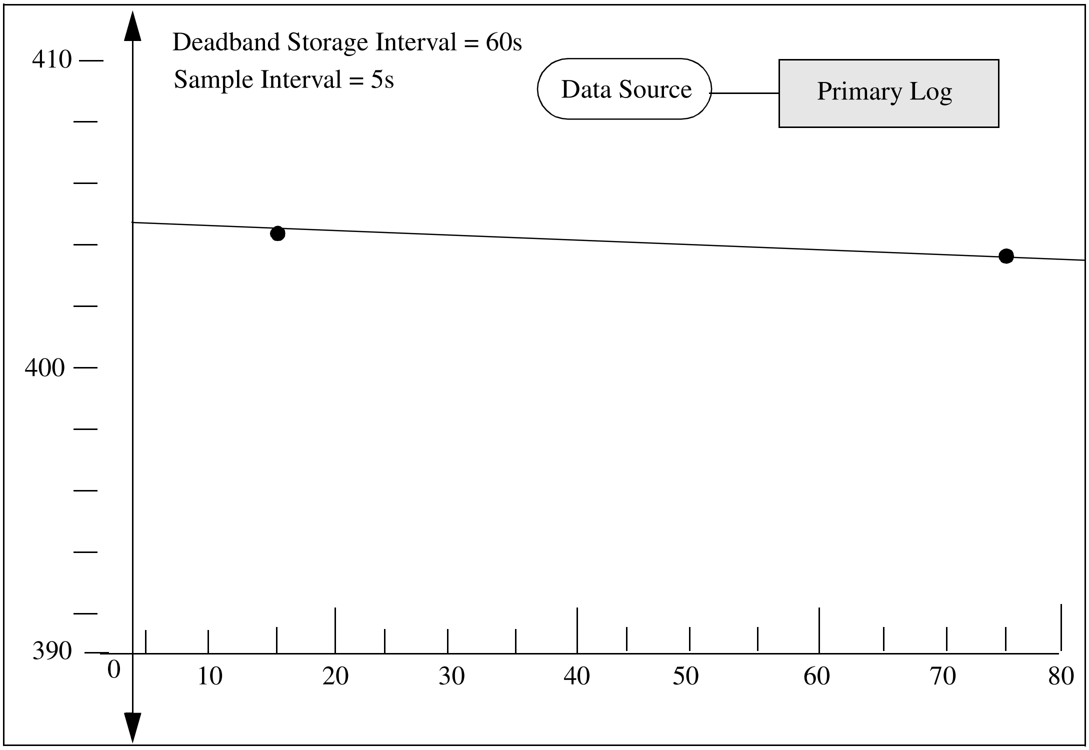


Figure 3-49. Storing Calculated Values w/Data Compaction

Table 3-2. Storing Calculated values w/Compaction

Interval	**Value**	**Calculated Average**	**Stored**	**Reason**
0	406.7	404.3	Yes	First calculated average
5	404.5			
10	406.2			
15	399.8.			
20	394.9	403.0	No	No significant change in slope between 1st & 2nd calculation
25	409.0			
30	404.2			
35	404.0			
40	402.8	403.4	No	No significant change in slope between 2nd & 3rd calculation
45	403.3			
50	403.2			
55	404.1			
60	404.1	403.7	Yes	Deadband storage interval (60s in this example) elapsed since last stored sample (0s)
65	403.8			
70	404.1			
75	402.9			

3.2.9 File Storage

Log entries can be stored directly in files maintained by History. There are two types of file storage as determined by the STORAGE_TYPE log attribute:

- TYPE1 = file storage @ 24 bytes per entry
- TYPE2 = file storage @ 4 bytes per entry

Section 3.2.9.1, Comparison of TYPE1 and TYPE2 File-based Logs provides guidelines for selecting whether TYPE1 or TYPE2 logs are more appropriate for your application.

For details regarding disk space requirements, see Section 3.2.9.2, Disk Requirements for File-based Numeric Logs.

3.2.9.1 Comparison of TYPE1 and TYPE2 File-based Logs

The default storage type is TYPE1 (24-byte). The 24-byte entry provides the full range of status and related information for each entry. When you select the TYPE2 storage type, each entry requires less storage space; however, you sacrifice some of the status information provided for TYPE1 entries as well as some accuracy with regard to time stamps. Also, for TYPE2 entries, if a value is modified, the original value is not saved, and no status bits are set to indicate that the value has been modified.

TYPE2 storage type may be appropriate for logs that have fast sample rates (one minute or faster), if you determine that the degree of inaccuracy with regard to time stamp is small enough to be acceptable.

The full range of information provided for TYPE1 entries is described below. The differences between TYPE1 and TYPE2 entries for each category of information are described where applicable.

Time Stamp

Each data entry in a log is time stamped. This stamp consists of two different times:

- Universal Time (UTC) Coordinate time stamp is Greenwich Mean Time. It is not affected by the time zone nor daylight savings time. This gives consistency throughout all external changes such as Daylight Savings Time and any system resets that affect the Local time.
- Local Time is the actual time of day dependent upon time zone and daylight savings time.

Data Value

The data value format depends upon the log type. Numeric logs can hold integer and decimal numbers.

Entry Status

The entry status information is a bit set. Each bit provides status information regarding the log entry as described in Table 3-3. Note that while the complete range of information is stored for TYPE1 logs, TYPE2 logs do not store all status bits. The absence of a status value indicates that the data is good.

Table 3-3. Description of Bit Set for Entry Status

Bits in HsLogStatusType	Bit number	Description	TYPE1	TYPE2
noData	1	No data was collected and stored for the entry at this time. Probably due to communications problems between History and the data source.	Yes	Yes
badData	2	Value stored in this entry has bad data. Probably due to a calculation that contains No Data entries in it.	Yes	Yes
entryArchived	3	Not used.	N/A	N/A
valueInAlarm	4	Not used.	N/A	N/A
alarmPriority0	5	Not used.	N/A	N/A

Table 3-3. Description of Bit Set for Entry Status

Bits in HsLogStatusType	Bit number	Description	TYPE1	TYPE2
alarmPriority1	6	Not used.	N/A	N/A
timeQuality	7	Not used.	N/A	N/A
timeChange	8	Not used.	N/A	N/A
sourceTimeStamp	9	Time stamp in the entry was obtained with the value collected. Values collected by History usually use the time stamp from the source.	Yes	No
timeStampAdded	10	History generated the time stamp. This is generally done for No Data entries.	Yes	No
_dummy0	11 - 12	Not used.	N/A	N/A
userEntered	13	User entered the data using the AddNumericValueEntry operation.	Yes	Yes
modifiedData	14	When requesting current values from History, this is set if the user modified the original value. If a value has been modified, the No Data and Bad Data bits can not be set. It is assumed the modified value is correct.	Yes	No
modifiedAvailable	15	When requesting original values from History, this is set if the user modified the original value. It indicates that a modified value is available.	Yes	No
_dummy1	16	Not used.	N/A	N/A
firstEntry	17	This bit is set when a log is activated. It does not represent a real value (No Data will also be set). It is a place holder in the log that indicates when the log was activated.	Yes	Yes
lastEntry	18	This bit is set in the status of the last entry collected when a log is deactivated.	Yes	Yes

Table 3-3. Description of Bit Set for Entry Status

Bits in HsLogStatusType	Bit number	Description	TYPE1	TYPE2
valueInterpolated	19	Value returned by History was interpolated during retrieval. It is not a value that is physically stored in the data base.	Yes	Yes
_dummy2	20	Not used.	N/A	N/A
sequenceNum0	21	These bits are used to indicate the sequence number of the log the entry was retrieved from. For example, if the entries were retrieved from log $HSTIC100,VALUE-5-o, then sequenceNum0 and sequenceNum2 would be set.	Yes	No
sequenceNum1	22		Yes	No
sequenceNum2	23		Yes	No
sequenceNum3	24		Yes	No
sequenceNum4	25		Yes	No
TTDValue (Master)	26	This bit is set if the entry was returned from TTD.	Yes	Yes
CCFValue (MOD 300)	27	This bit is set if the entry was returned from CCF Trend Buffer.	Yes	Yes
_dummy3	28 - 32	Not used.	N/A	N/A

Object Status

Object status is the status of the data source when the value was sampled. The format of the status (what is represented and how) is dependent upon the data source object type. When a secondary log is created, the object status is obtained from the last primary log entry used in the calculation.

Object status is not stored for TYPE2 logs.

3.2.9.2 Disk Requirements for File-based Numeric Logs

For file-based logs, the actual disk space may be greater than the calculated capacity if the log is very small. For TYPE1 (24-byte) logs, the minimum capacity allocated on disk is 84 entries. For TYPE2 (4-byte) logs, the minimum space allocated on disk is 500 entries.

Each TYPE1 log requires about 24 bytes per entry. To calculate exact file size for TYPE1:

- #data_blocks = (#entries+83)/84
- #index_blocks = (#data_blocks+253)/254
- file size = (#data_blocks+#index_blocks)*2048

Each TYPE2 log requires about 4 bytes per entry. To calculate exact file size for TYPE2:

- #data_blocks = 1+ (#entries+499)/500
- #index_blocks = (#data_blocks+253)/254
- file size = (#data_blocks+#index_blocks)*2048

3.3 Building History Logs with Text Files

Typically, the History database is predominantly made up of synchronous numeric logs. The dbConfig utility helps you configure large numbers of numeric logs through text files. This method is quicker than the History Builder *Save As* method.

The basic procedure is described in Section 3.3.1, An Example Application of dbConfig. How to manage database changes after initial configuration and loading is described in Section 3.3.2, Dumping and Loading Changes. For details regarding all dbConfig options, see Section 3.3.3, dbConfig.

Do not start by creating text files. Create your History objects via the History Builder first. The recommended work-flow is:

1. Using the History Builder, configure one composite log for each basic log configuration that your application requires. These composite logs will be used as templates to create the additional numeric logs in the database.
2. Use a text editor such as Notepad or a spreadsheet tool such as Microsoft Excel to create tag lists. These tags are the actual Composite Log Names for the numeric logs being built from the composite log templates. Create a separate list (text file) for each composite log template created in step 1 above.
3. Use dbConfig to dump the partial history database (consisting of template logs) into a text file. You can perform this step either from a command line interface (Windows NT Command Prompt), or from the GUI-based History Utility.
4. Use dbConfig to create a text file with *like* statements for each composite log template. These files contain *like* statements that specify which tag list to use, the composite log template to use, and where to put the *like* statement file. This step is performed via the Windows NT Command Prompt.
5. Use dbConfig to parse and check the dump file and like files. This step is performed via the Windows NT Command Prompt.
6. Drop/recreate the database. This is optional. If you do not perform this step, the new logs you create will be appended to any existing logs in the current History database. If you want to start with a fresh (empty) database, then you must drop and recreate the History database.

7. Use dbConfig to load the complete History database from the text files. You can perform this step either via Windows NT Command Prompt, or the GUI-based History Utility.

   NOTE

   - Stopping History before you load the new configuration helps the load run faster. You are not required to stop History before you load the new configuration; however, it may be worth while if the new configuration is large.
   - Also, you will be required to stop and then re-start History to make the new logs accessible.

   dbConfig checks the syntax of the loaded files and verifies the configuration information. If there are no errors, dbConfig creates the history objects defined in the text files.

8. Create a backup copy of your database.

9. Stop and then re-start History to make the new logs accessible.

3.3.1 An Example Application of dbConfig

The following procedure is based on three composite log structures as described in:

- Section 3.2.4.1, Simple Numeric Logs for Synchronous Process Data
- Section 3.2.4.2, Secondary Logs for Different Resolution/Time Span Views
- Section 3.2.4.3, Primary and Secondary Logs with Calculations

3.3.1.1 Use the History Builder to Configure a Database Template

To build the partial database for this example, you must use the History Builder to build the three composite log structures described in Section 3.2.4, Adding Numeric Logs to the Composite Log.

NOTE

For Composite Log Name, do not use an actual data source name. Instead use a label that identifies this composite log as a template, for instance: `InstTemplate`, `AveTemplate`, and `MinMaxTemplate`.

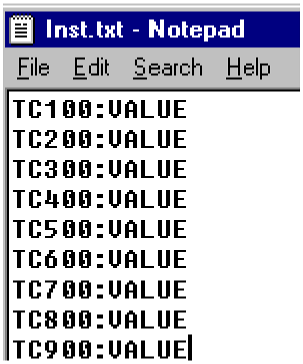
3.3.1.2 Create Tag Lists for Additional Logs

The dbConfig utility reads text files to get the names of all the numeric logs to be created in the History database. One text file is required for each template composite log configured in the History Builder. In this case, three text files are required:

- `Inst.txt` provides the names for all logs created with `InstTemplate` (Instantaneous data collection).
- `Ave.txt` provides the names for all logs created with `AveTemplate` (1-minute and 1-hour averages).
- `MinMax.txt` provides the names for all logs created with `MinMaxTemplate` (Minimum and Maximum Values.

These text files can be created with a spreadsheet application, or a text editor such as notepad. Each line specifies one tag name.

Enter the OPC tag as it is specified in the OPC server, Figure 3-50.



```
TC100:VALUE
TC200:VALUE
TC300:VALUE
TC400:VALUE
TC500:VALUE
TC600:VALUE
TC700:VALUE
TC800:VALUE
TC900:VALUE
```

Figure 3-50. Example Tag List Text File

3.3.1.3 Read History Database into a Text File

The partial History database configured with the History Builder must be dumped to a text file. This example application uses the command line method via the Windows NT Command Prompt. You may use the GUI-based History Utility if you prefer. For details, see Section 5.2.15.1, Dump Configuration.

Access to the `dbConfig` function is via the `%HS_HOME%` environment variable which points to the directory where the dbConfig function resides. To access the dbConfig function, first open the Windows NT Command Prompt. To do this, from the task bar choose **Start>Programs>Command Prompt**. Then, in the Command Prompt window, change the directory as follows:

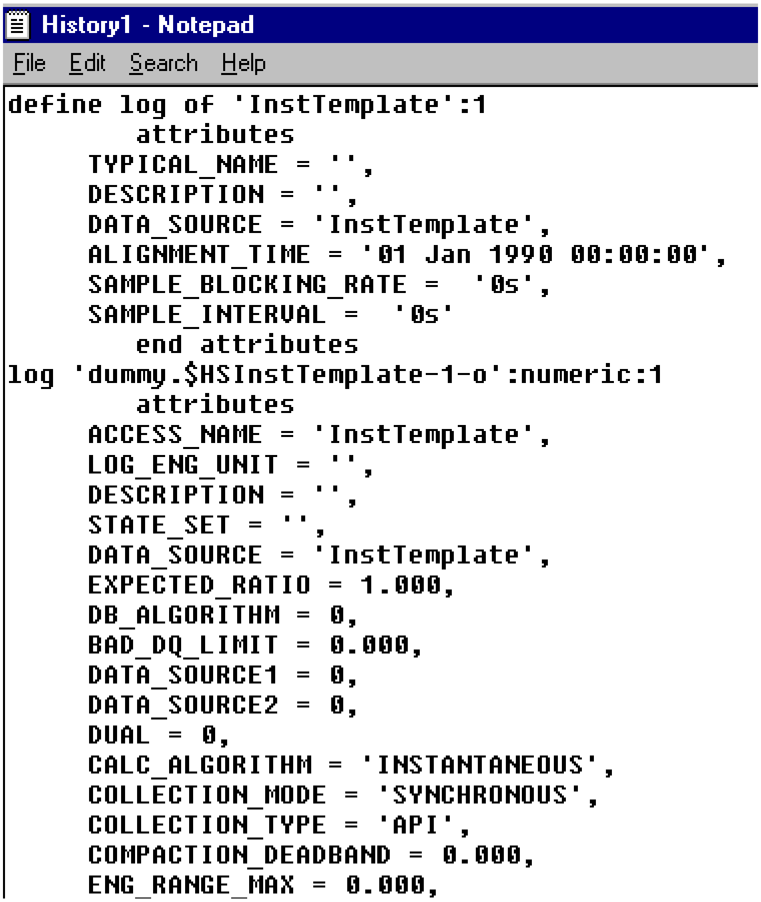
```
c:\> cd “%HS_HOME%\bin”
```

Use `dbConfig` with the `-d` option to read the history configuration out of the History database and write this configuration to a text file. You must specify the file path in the dbConfig command.

For example, to dump the database to a text file named `History1` in the `TEMP\HistoryDB` directory, enter:

```
dbConfig -d > \TEMP\HistoryDB\History1
```

Once you have created a text file using dbConfig, you can open the file for viewing and editing using your text editor. The file contains text-based log definitions for all logs currently configured in the History database. A partial listing of `History1` is shown in Figure 3-51.



```
define log of 'InstTemplate':1
        attributes
     TYPICAL_NAME = '',
     DESCRIPTION = '',
     DATA_SOURCE = 'InstTemplate',
     ALIGNMENT_TIME = '01 Jan 1990 00:00:00',
     SAMPLE_BLOCKING_RATE =  '0s',
     SAMPLE_INTERVAL =  '0s'
        end attributes
log 'dummy.$HSInstTemplate-1-o':numeric:1
        attributes
     ACCESS_NAME = 'InstTemplate',
     LOG_ENG_UNIT = '',
     DESCRIPTION = '',
     STATE_SET = '',
     DATA_SOURCE = 'InstTemplate',
     EXPECTED_RATIO = 1.000,
     DB_ALGORITHM = 0,
     BAD_DQ_LIMIT = 0.000,
     DATA_SOURCE1 = 0,
     DATA_SOURCE2 = 0,
     DUAL = 0,
     CALC_ALGORITHM = 'INSTANTANEOUS',
     COLLECTION_MODE = 'SYNCHRONOUS',
     COLLECTION_TYPE = 'API',
     COMPACTION_DEADBAND = 0.000,
     ENG_RANGE_MAX = 0.000,
```

Figure 3-51. Example Dump File, Partial View

3.3.1.4 Generate *Like* Files

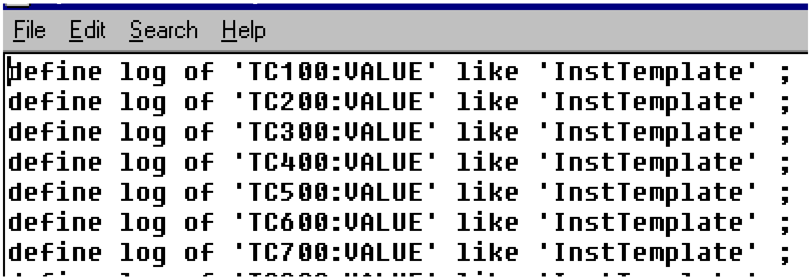
Use dbConfig with the -t, -l and > options to generate a text file with *like* statements. Use the Windows NT Command Prompt to enter the dbConfig specification.

One text file is required for each of the composite log templates configured via the History Builder. The *like* statements specify which tag list to use, the composite log template to use, and where to put the *like* statement file. For example:

```
dbConfig -t \TEMP\TagLists\Inst.txt -l InstTemplate >
\TEMP\LikeLists\Inst.like
```

- After the `-t` option, enter the name of the text file that contains the tag list to read. For example, the statement above specifies Inst.txt in the `\TEMP\TagLists` directory.
- After the `-l` option, enter the name of composite log template defined via the History Builder that the numeric logs will be *like* . In this case, the numeric logs will be *like* `InstTemplate`.
- After the > character, enter the directory and name of the text file that will contain the like statements. In this case the *like* statements will be written to Inst.like in the `\TEMP\LikeLists` directory.

You can open the *like* lists with your text editor. A partial listing of Inst.like is shown in Figure 3-52.



```
define log of 'TC100:VALUE' like 'InstTemplate' ;
define log of 'TC200:VALUE' like 'InstTemplate' ;
define log of 'TC300:VALUE' like 'InstTemplate' ;
define log of 'TC400:VALUE' like 'InstTemplate' ;
define log of 'TC500:VALUE' like 'InstTemplate' ;
define log of 'TC600:VALUE' like 'InstTemplate' ;
define log of 'TC700:VALUE' like 'InstTemplate' ;
```

Figure 3-52. Example, Text File with "Like" Statements

NOTE

A composite log typical is created for each *like* file when those files are loaded as described in Section 3.3.1.6, Configure the History Database from the Text Files. The composite log typical name is derived from the template composite log, for instance: Logs_Like_LogTemplate_Node_1. All logs created from the template composite log are included in the composite log typical.

3.3.1.5 Generate a Report for dbConfig Output

As an option, you can use dbConfig generate a report for the dump file and like files. For example:

```
dbConfig \TEMP\HistoryDB\History1
\TEMP\LikeLists\Inst.like \TEMP\LikeLists\Ave.like
\TEMP\LikeLists\MinMax.like > \TEMP\LogFile
```

An example report is shown in Figure 3-53. To direct the report to a text file use the > option as shown above. If you do not use this option, the output is directed to the screen. The report has two parts:

- The top part, labeled Summary of Oracle/Disk Requirements for Runtime Logs, is an overall summary for all History nodes.
- The bottom part, labeled Summary of Oracle/Disk Requirements by Node, is a summary of the Oracle/Disk requirements for a specific node. In a single node configuration, this part is essentially a repeat of the first part.

The contents of this file are described in Table 3-4.

Table 3-4. dbConfig Output Report

INFORM_HS_RUNTIME TABLESPACE	Disk space required for History objects stored in the Inform_HS_Runtime tablespace. Since there are no Oracle-based logs in this example, the Oracle/disk requirement for this component is 0.00 Meg.
HS_INDEXES TABLESPACE	Disk space required for the HS_Indexes tablespace. Since this is related to Oracle-based logs, the Oracle/disk requirement for this component is also 0.00 Meg.
TOTAL ORACLE DISK USAGE	Total disk space required for Oracle tables. This is a sum of the INFORM_HS_RUNTIME and HS_INDEXES fields.
NUMBER OF FILES	Total number of file-based numeric logs.
ACTUAL DISK USAGE FOR FILES	
TOTAL CAPACITY	Sum of log capacity for all logs.
PRIMARY SAMPLE RATE	Average sample rate for all synchronous logs.

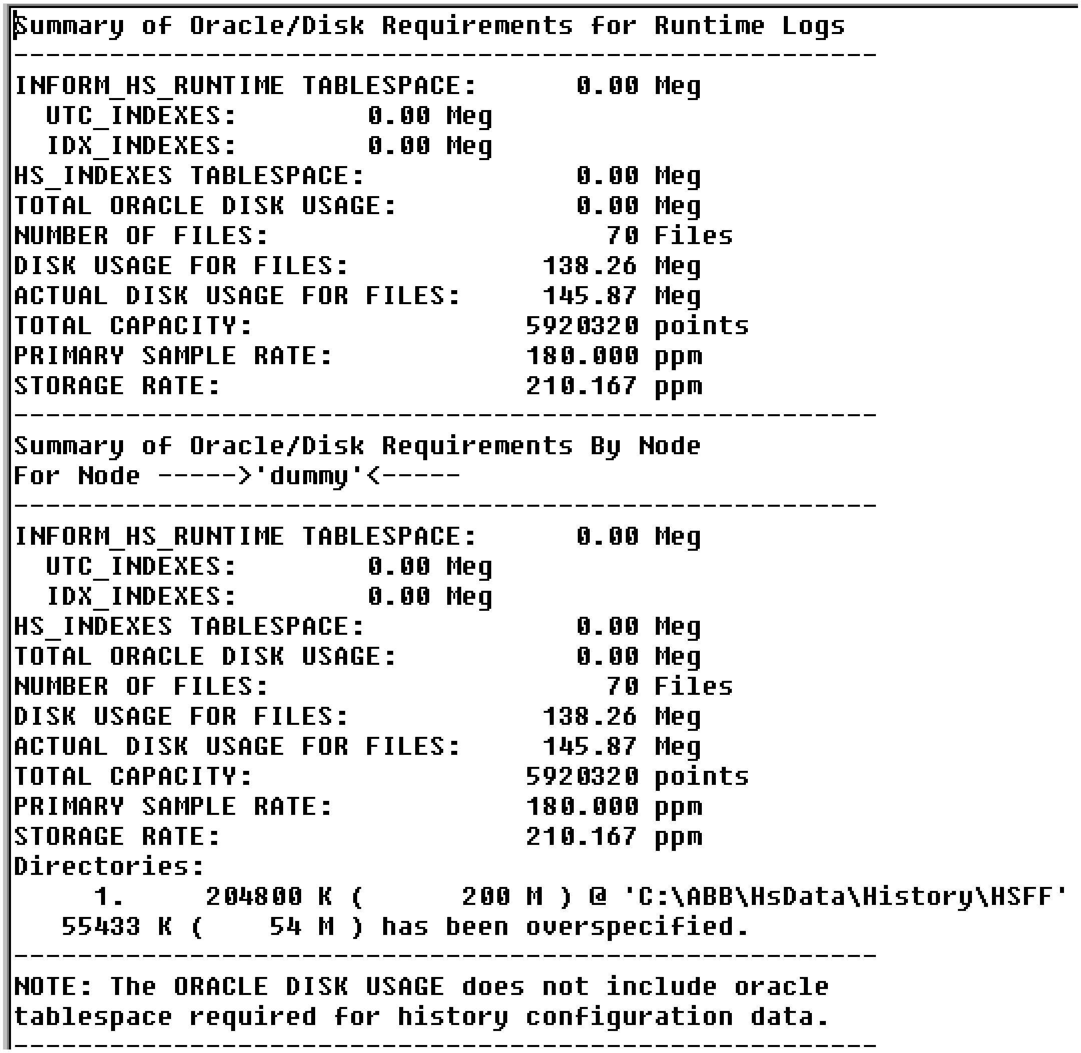
Table 3-4. dbConfig Output Report

STORAGE RATE	Average storage rate for all synchronous logs.
Directories	Allocation of disk space in directories configured for file-based storage. In this example, one directory has been configured: C:\ABB\HsData\History\HSFF has been allocated 200 Meg. The line directly below the Directories list indicates whether or not enough disk space has been allocated for the current configuration. In this case, the disk allocation is 54 Meg. more than is required by the History database configuration 200 Meg. - 145.87 Meg. = 54 Meg.

If the dbConfig output indicates History resources are overloaded by the configuration, you may be able to fix the problem by modifying the History database using one or more of the following methods:

- change the period of a numeric log
- decrease the points per minute sample from the control network
- increase the amount of disk space allocated to History

These changes are made to the History database template, and should be relatively easy to make. If your functional requirements are fixed and can not be changed, more hardware will be needed (a second History node, and/or faster machine).



```
Summary of Oracle/Disk Requirements for Runtime Logs
------------------------------------------------------
INFORM_HS_RUNTIME TABLESPACE:      0.00 Meg
  UTC_INDEXES:        0.00 Meg
  IDX_INDEXES:        0.00 Meg
HS_INDEXES TABLESPACE:             0.00 Meg
TOTAL ORACLE DISK USAGE:           0.00 Meg
NUMBER OF FILES:                     70 Files
DISK USAGE FOR FILES:            138.26 Meg
ACTUAL DISK USAGE FOR FILES:     145.87 Meg
TOTAL CAPACITY:                 5920320 points
PRIMARY SAMPLE RATE:            180.000 ppm
STORAGE RATE:                   210.167 ppm
------------------------------------------------------
Summary of Oracle/Disk Requirements By Node
For Node ----->'dummy'<-----
------------------------------------------------------
INFORM_HS_RUNTIME TABLESPACE:      0.00 Meg
  UTC_INDEXES:        0.00 Meg
  IDX_INDEXES:        0.00 Meg
HS_INDEXES TABLESPACE:             0.00 Meg
TOTAL ORACLE DISK USAGE:           0.00 Meg
NUMBER OF FILES:                     70 Files
DISK USAGE FOR FILES:            138.26 Meg
ACTUAL DISK USAGE FOR FILES:     145.87 Meg
TOTAL CAPACITY:                 5920320 points
PRIMARY SAMPLE RATE:            180.000 ppm
STORAGE RATE:                   210.167 ppm
Directories:
     1.     204800 K (      200 M ) @ 'C:\ABB\HsData\History\HSFF'
   55433 K (    54 M ) has been overspecified.
------------------------------------------------------
NOTE: The ORACLE DISK USAGE does not include oracle
tablespace required for history configuration data.
------------------------------------------------------
```

Figure 3-53. Example Output from dbConfig

3.3.1.6 Configure the History Database from the Text Files

If no problems are indicated in the dbConfig output, use the History Utility to load the database from text files. This example application uses the command line method via the Windows NT Command Prompt. You may use the GUI-based History Utility if you prefer. For details, see Section 5.2.15.2, Load Configuration.

The load can be done with History running; however, it will take longer. Also, History must be stopped and then re-started anyway to enable the new logs to collect data. Instructions for starting and stopping History are provided in the *Advant Workplace Administrator's Guide*.

Dropping the Current Database

As an option, you can use the `hsDBMaint` utility to drop the current database and create an empty database *before* you load the database from text files. Do this only when your current database contains bad or useless data as a result of testing or experimenting. Refer to Section 5.2.16, Create or Drop a Product Database for instructions on dropping and creating a database.

Before you use hsDBMaint to drop and create a database, you must first stop the History Server, and then restart the History server when you are finished. Refer to the *Advant Workplace Products Administrator's Guide* for details on how to do this.

If necessary, use hsDBMaint Extend Tablespace function to extend tablespace, and Directory Maintenance function to increase the quota for file-based logs. The quota should slightly exceed the numbers provided by dbConfig.

Loading the Database from Text Files

Specify the filenames for the text file created by the dump command and the text files with *like* statements.The node specification (dummy) is optional. For example:

```
dbConfig -l dummy \TEMP\HistoryDB\History1
\TEMP\LikeLists\Inst.like \TEMP\LikeLists\Ave.like
\TEMP\LikeLists\MinMax.like
```

If this load fails because of disk space problems or Oracle tablespace problems, just repeat the procedure after you correct the problems.

3.3.1.7 Restarting History

History must be stopped and then re-started to enable the new logs to collect data. Instructions for starting and stopping History are provided in the *Advant Workplace Administrator's Guide*.

3.3.2 Dumping and Loading Changes

After loading a database using dbConfig, you can modify the configuration using the History Builder. For instance you may adjust deadband compaction ratios on some logs. Any instance where a log attribute definition differs from the attribute definition of the corresponding composite log typical is considered a change.

When you make such changes via the History Builder, it is recommended that you save (dump) these changes to a text file. Then, if you need to restore your database for some reason, you can use dbConfig to restore the original configuration, and then restore any saved changes that you dumped to the text file.

Dumping Changes to a Text File

To save changes to a text file, use dbConfig with the -d and -c options. This dumps just the database changes to a text file. For example, the following statement dumps database changes to the text file `dbChanges`

```
dbConfig -d -c dbChanges
```

The text file is shown in Figure 3-54.

```
Changes for Node nodeA
Differences for NLog $HSTAG0001,MEASURE-3-o
SAMPLE_INTERVAL =  '2m'
```

Figure 3-54. Example, Changes Text File

Restoring the Database with Changes

To restore the database with changes, use dbConfig with the -l and -c options. This first loads the original database from the specified text file, and then loads the changes from the changes text file. The file name for the changes file must follow directly after the -c option. The path specification for the -l option must be at the end of the command (after all other options). For example:

```
dbConfig -l -c dbChanges /tmp/History1 /tmp/100.like
```

If this load fails because of disk space problems or oracle tablespace problems, just repeat the procedure after you correct the problems.

3.3.3 dbConfig

Access to the `dbConfig` function is via the `%HS_HOME%` environment variable which points to the directory where the dbConfig function resides. To access the dbConfig function, first open the Windows NT Command Prompt. To do this, from the task bar choose **Start>Programs>Command Prompt**. Then, in the Command Prompt window, change the directory as follows:

```
c:\> cd "%HS_HOME%\bin"
```

Use dbConfig to:

- Dump a History Database to a Text File
- Create a Text File with Like Statements
- Check Syntax
- Load (Create) a Complete Database from Text Files
- Dump History Database Changes to a Text File
- Restore a Complete Database with Changes from Text Files

Dump a History Database to a Text File

Use `dbConfig` with the `-d` option to read the history configuration out of the History database and write this configuration to a text file. To save this information to a file, you must specify the file in the dbConfig command.

For example, to dump the database to a text file named `History1` in the `TEMP\HistoryDB` directory, enter:

```
dbConfig -d > \TEMP\HistoryDB\History1
```

Create a Text File with *Like* Statements

Use `dbConfig` with the `-t`, `-a`, and `-l` options to create a text file with "like" statements that specify how to configure the bulk of the numeric logs based on template logs configured via the History Builder. For example:

```
c:\dbConfig -t \tmp\group1.tags -a MEASURE -l LOG1 >
\tmp\group1.like
```

This creates a text file of 'like' statements for tags listed in the /tmp/group1.tags text file, using the MEASURE attribute, and the LOG1 composite log template from the History database:

Check Syntax

Use dbConfig without options to check file syntax and perform dependency checking equivalent to checks done in the history builder. For example:

```
dbConfig \TEMP\HistoryDB\History1
\TEMP\LikeLists\Inst.like \TEMP\LikeLists\Ave.like
\TEMP\LikeLists\MinMax.like
```

If database errors are found in the dbConfig file (for instance the value of a high limit attribute is set higher than the high-high limit attribute), the file will not be able to be loaded unless the file is corrected, or the affected logs are deleted from the dbConfig file.

Load (Create) a Complete Database from Text Files

Use `dbConfig` with the `-l` option `to` load the configuration information from the dbConfig text file into the local History database. Specify the filenames for the text file created by the dump command and the text files with *like* statements.The node specification (dummy) is optional. For example:

```
dbConfig -l dummy \TEMP\HistoryDB\History1
\TEMP\LikeLists\Inst.like \TEMP\LikeLists\Ave.like
\TEMP\LikeLists\MinMax.like
```

This uses the Dump file named `History1` in `\TEMP\HistoryDB`.

Usually, dbConfig is used to load data into a clean history database (that is, no other objects have been configured for that node), or to add more logs to an existing database. If objects that already exist in the database are present in the file being loaded, dbConfig will report an error and continue creating other objects in the file. The existing object is not affected.

As an option, you can use the `hsDBMaint` utility to drop the current database and create an empty database *before* you load the database from text files. Do this only when your current database contains bad or useless data as a result of testing or experimenting. Refer to Section 5.2.16, Create or Drop a Product Database for instructions on dropping and creating a database.

Before you use hsDBMaint to drop and create a database, you must first stop all Advant OCS software on the History node, and then restart the History node when you are finished. Refer to the *Advant Workplace Products Administrator's Guide* for details on how to do this.

If necessary, use hsDBMaint Extend Tablespace function to extend tablespace, and Directory Maintenance function to increase the quota for file-based logs. The quota should slightly exceed the numbers provided by dbConfig.

Dump History Database Changes to a Text File

Use `dbConfig` with the `-d` and -c options to read configuration changes out of the History database and write just the changes to `stdout` (standard output file). To save this information to a file, you must specify the file in the dbConfig command. For example:

```
c:\tbd\dbConfig -d testnode1 -c<filename>
```

Restore a Complete Database with Changes from Text Files

If you need to restore your database for some reason, you can use dbConfig to restore the original configuration (before changes), and then restore any changes that were made to the original configuration (assuming those changes were saved). For example:

```
c:\tbd\dbConfig -l testnode1
-ctemplateAChanges.db \tmp\dumpfile
```

General Syntax and Options

The syntax for dbConfig is as follows:

```
dbConfig [options] <filename> <filename2>
```

The options are as follows:

`-d`	Use this option to dump a History database and redirect it to a text file. This option can not be used in combination with -t. For an example, see Section 3.3.1.3, Read History Database into a Text File.
`-t`	Use this option to create a text file with "like' statements that will be used to configure the bulk of the numeric logs in the History database based on the template logs configured via the History Builder. For an example, see Section 3.3.1.4, Generate Like Files.
`-a`	Use this option in conjunction with the -t option to specify the tag attribute for the 'like' statement. For an example, see Section 3.3.1.4, Generate Like Files. DO NOT use this option if the attribute is already specified in the tag list.
`-l`	Use this option in conjunction with the -t option to specify the composite log template for the 'like' statement. For an example, see Section 3.3.1.4, Generate Like Files
`-n` *`nodename`*	Identifies the history node that is being dumped as *`nodename`*. Nodename can be any user defined text and is used by dbConfig to identify the correct objects for a node being loaded. Defaults to *dummy*. This option is only valid in combination with the `-d` option.
> *filename*	Identifies file that history configuration text file is being redirected to.
`-l nodename`	Identifies the history node that is being loaded as *`nodename`*. Nodename must be the nodename provided when the node was dumped using dbConfig. The `-l` option tells dbConfig to actually load the file into the database. If `-l` is not provided, no data is loaded; however, the data is checked for validity. For an example, see Section 3.3.1.6, Configure the History Database from the Text Files.

`-v`	Print out the configuration information in the dbConfig file. This option will print each object that will be created in the database.
`-a`	Print out the attributes for each object. This option should be used with the `-v` option in order to get detailed attribute information for each object printed with the `-v` option.
-c	Use this option in conjunction with the -d option to dump database changes to a text file, or with the -l option to load changes to a restored database. For examples, see Section 3.3.2, Dumping and Loading Changes.

3.4 Starting and Stopping Data Collection

You start and stop data collection by activating or deactivating logs. You start and stop logs on an individual basis, or start/stop all logs in a composite log simultaneously. See Section 3.2.7, Starting/Stopping Logs.

Chapter 4 Operation

4.1 Getting Started

You can retrieve historical data using Advant Trends. For guidelines, see:

Section 4.2, When to Use Composite Log Name or Numeric Log Name

Section 4.3, Retrieval for Advant Trend Displays

Section 4.4, Criteria for Selection for Seamless Retrieval

Section 4.5, Interpolation

NOTE

Data that has been collected but not yet stored to disk by History (due to the sample blocking rate) is not available for retrieval.

4.2 When to Use Composite Log Name or Numeric Log Name

There are two ways to specify which log to retrieve data from. The Composite Log Name is the name of the data source that you assign to a log when you configure it in the History Builder. For instance TC100:VALUE.

By virtue of the hierarchical log structure of History logs, there can be more than one log using the same data source. To distinguish multiple logs with the same data source, History automatically assigns a unique Numeric Log Name to each primary and secondary log within a composite log structure. The Numeric Log Name is derived from the Composite Log Name, and has a prefix and suffix appended as shown in Figure 4-1.

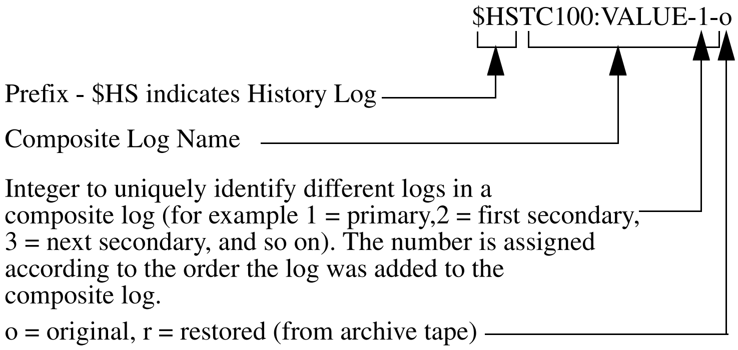

$HSTC100:VALUE-1-o
Prefix - $HS indicates History Log
Composite Log Name
Integer to uniquely identify different logs in a composite log (for example 1 = primary,2 = first secondary, 3 = next secondary, and so on). The number is assigned according to the order the log was added to the composite log.
o = original, r = restored (from archive tape)

Figure 4-1. Log Name

You can use the Composite Log Name to specify which log to retrieve data from when you know the data source but do not know the Numeric Log Name. When you use the Composite Log Name, History uses the seamless retrieval criteria (described later in this section) to select the most appropriate log in the data source's log hierarchy. Generally, unless other criteria are specified, seamless retrieval selects the log that has INSTANTANEOUS or AVERAGE as the calculation algorithm, and has a log period that provides the best coverage of the requested time period.

Use the Numeric Log Name when you do not want to rely on the seamless retrieval criteria to select the log, and you know the name of the primary or secondary log that stores the required data. For instance, you can use the Numeric Log Name if you want the log that uses the STANDARD DEVIATION algorithm, and know that this data is stored in $HSTC100:VALUE-3-o.

4.3 Retrieval for Advant Trend Displays

The Advant Trends software supports viewing of History data on standard trend displays. All you have to do to configure these displays is to specify the variables you wish to see on them.

Trend displays are built with data from the trend buffer, unless a data request is entered to extend the X Axis back in time beyond the capacity of the buffer, or no trend is configured. Such a request can be entered via the Modify X Axis dialog as described in the *Advant Trend User's Guide*. In this case, the appropriate history log is used to build most of the trend, and the gap from the beginning of the log to present (due to delay caused by storage interval and sample blocking rate) is filled by the trend buffer.

To extend the X axis, there must be one or more history logs configured for the loop whose trend is being displayed. If the log name is used in the data request, History knows exactly which log to retrieve data from. If the access name is used, the seamless retrieval criteria is used to select the most appropriate log.

The following example illustrates how a trend is built according to a data request entered via the Modify X Axis dialog.

A data request is entered in the Modify X Axis dialog for 8 hours worth of data from TC100:VALUE (access name). A History log with a composite log hierarchy as shown in Figure 4-2 is configured in the History database for TC100:VALUE.

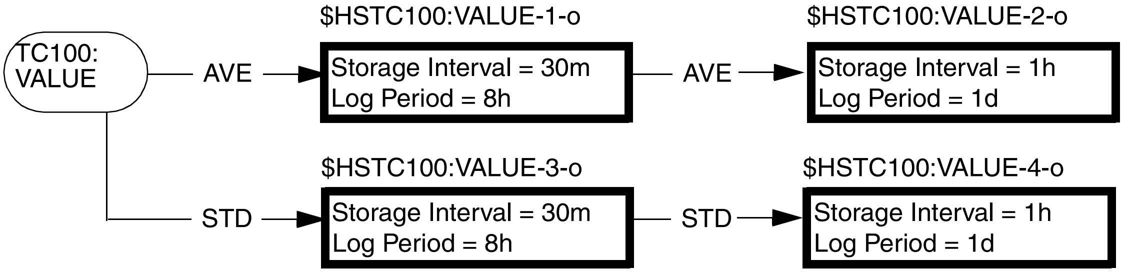


Figure 4-2. Example, Log Hierarchy

The log that most closely fits the data request is $HSTC100:VALUE-1-o since the log period is eight hours and the calculation algorithm is AVERAGE. Since the storage interval is 30 minutes, there is the potential to lack as much as 30 minutes of the most current data, or possibly more since the gap is dependent on the blocking rate. This gap is filled in as the data becomes available.

4.4 Criteria for Selection for Seamless Retrieval

Seamless retrieval uses the following criteria to search the History database and select a log when a request for history data is issued:

1. Sort logs by percentage of the time period of the request that the log covers, and choose the set of logs that are closest in coverage.

   For instance, for a request for 24 hours, logs may be sorted as follows in descending order of preference: 24-hour logs and 168_hour logs, 8-hour logs, and 1-hour logs. Since the 24-hour logs and 168-hour logs cover 100% of the requested time period, they are chosen.

2. Search the set chosen in step 1, and find all logs that use the specified calculation algorithm. Logs that do not use the specified algorithm are dropped from consideration.

   If the algorithm is WILDCARD (no algorithm specified), then the algorithms have the following order of preference:

   a. AVERAGE or INSTANTANEOUS

      These algorithms have equal preference. The first one found in the search is selected, and logs with any other calculation are dropped from consideration.

   b. MAX

   c. MIN

   d. SUM

   e. SUM OF SQUARES

   f. STANDARD DEVIATION

   g. NUM_OF_VALUES

   At this point the logs still under consideration have the same log period (closest to percent of coverage of requested time period), and calculation algorithm.

3. This set is now sorted, depending on the specified retrieval type. One log is selected according to retrieval type as follows:

   a. If the retrieval type is RAW, one log is selected in the following order of preference:

      1) Pick the log whose storage interval is equal to the time between points in request.

      2) Pick the log whose storage interval is greater than and closest to the time between points (so request can be completed in one response).

      3) Pick the log whose storage interval is less than and closest to the time between points (more points exist than buffer can hold, MORE_DATA status is returned).

   b. If the retrieval type is INTERPOLATED, one log is selected in the following order of preference:

      1) Pick the log whose storage interval is less than and closest to the time between points (so interpolation is more accurate).

      2) Pick the log whose storage interval is equal to the time between points in request.

      3) Pick the log whose storage interval is greater than and closest to the time between points.

   c. If retrieval type is DISPLAY, one log is selected according to retrieval type as follows:

      1) Pick the log whose storage interval is equal to the time between points in request. The request is changed to RAW unless the log uses Deadband compaction, in which case the request is changed to INTERPOLATE/AVE.

      2) Pick the log whose storage interval is greater than and closest to the time between points. The request is changed to INTERPOLATE/AVE.

      3) Pick the log whose storage interval is less than and closest to the time between points. The request is changed to INTERPOLATE/AVE.

4.5 Interpolation

History usually samples the values of numeric variables at set intervals. When data is requested at the interval it was stored, the software returns raw data. If data is requested at a different interval, the software returns an interpolated value. Figure 4-3 shows an example where data is saved every 10 seconds.

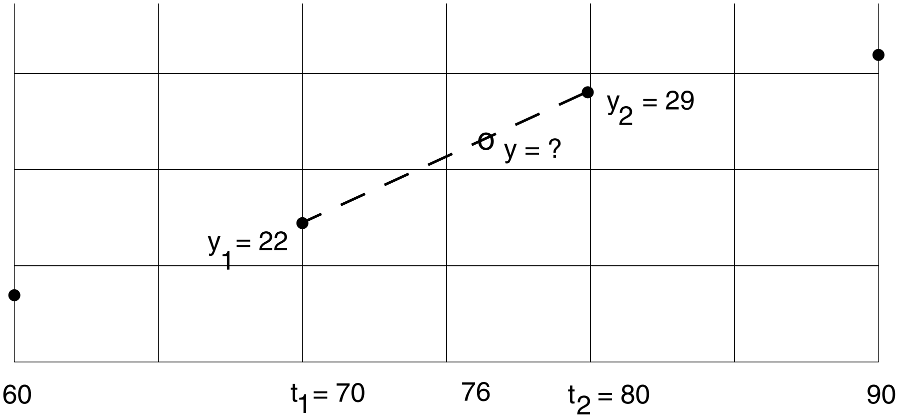


Figure 4-3. Numerical Data Retrieval by Interpolation

If the value is requested at second 70, the stored value 22 is returned. If the value is requested at second 80, the stored value 29 is returned. If the value is requested at second 76 (in between intervals where the value is stored), the stored values for seconds 70 and 80 are used to perform a mathematical interpolation. In this case, the value 26.2 is returned.

Chapter 5 Maintenance

The following maintenance topics are covered:

- Section 5.1, Preventive Maintenance
- Section 5.2, Database Maintenance Functions

5.1 Preventive Maintenance

Advant History is a part of the AdvaSoft for Windows NT software package. Perform any required maintenance according to the instructions in the *Advant Workplace Products Administrator's Guide*.

5.2 Database Maintenance Functions

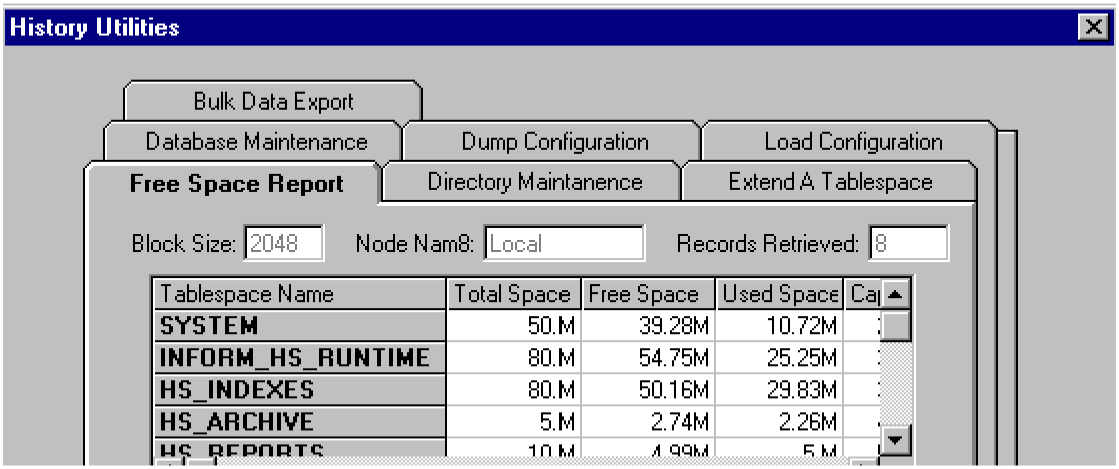
You can perform database maintenance functions either via the History Utilities GUI-based interface, or via the dbMaint command line interface.

Not all database maintenance functions are supported by the History Utilities.

- See Section 5.2.1, History Utilities for GUI-based Database Maintenance.
- See Section 5.2.2, hsDBMaint for Command Line Database Maintenance.

5.2.1 History Utilities for GUI-based Database Maintenance

The History Utility for GUI-based database maintenance is shown in Figure 5-1. To display this utility choose **Tools > History Utilities** from the History Builder menu bar.



Block Size: 2048 Node Nam8: Local Records Retrieved: 8

Tablespace Name	Total Space	Free Space	Used Space	Ca
SYSTEM	50.M	39.28M	10.72M	
INFORM_HS_RUNTIME	80.M	54.75M	25.25M	
HS_INDEXES	80.M	50.16M	29.83M	
HS_ARCHIVE	5.M	2.74M	2.26M	
HS_REPORTS	10.M	4.99M	5.M	

Figure 5-1. History Utilities for On-line Database Maintenance

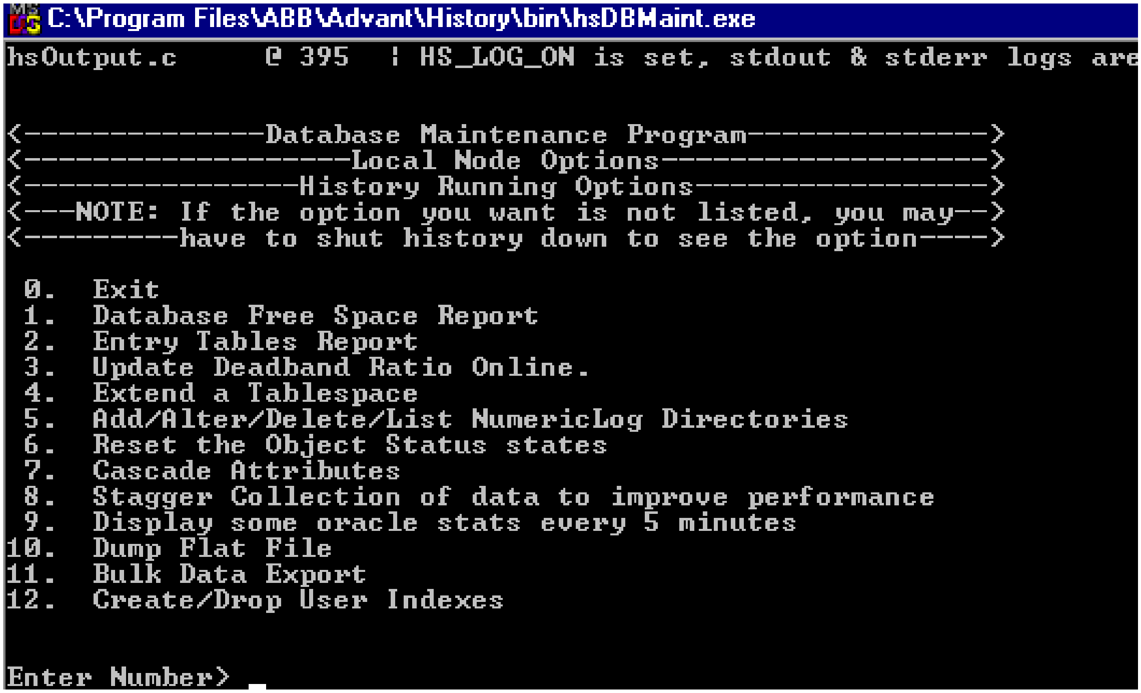
Refer to the following for details regarding the supported functions:

- Section 5.2.3, Database Free Space Report - to generate a report on free space in Oracle tablespaces on the History disk.
- Section 5.2.4, Entry Tables Report - to create a comprehensive list of all numeric logs, their current capacity, and other useful statistics.
- Section 5.2.5, Update Deadband Ratio On-line - to update deadband ratio and estimated log time period for logs that use deadband.
- Section 5.2.6, Extend Tablespace for History Files - to extend the tablespace in your system when you require greater storage capacity than provided by default Oracle tablespace sizes.
- Section 5.2.7, Maintain Directories for File-based Logs - to configure directories for file-based storage.

- Section 5.2.8, Reset Object Status States - to reset *attrsUpdated* and *userEntered* bits. These bits determine whether the collection function will update the trend presentation attributes, or whether user-entered values will be used. This is not applicable for OPC-type logs.
- Section 5.2.9, Cascade Attributes for Composite Logs - to propagate presentation, engineering units, and alarm attributes from the data source to secondary logs in the composite hierarchy (see Section 3.2.8.4, Alarm Attributes). This needs to be done when the attributes are configured for the primary log, but not the secondary logs. In order for deadband to work properly on the secondary log, the node must be re-started after the cascade is done.
- Section 5.2.10, Stagger Collection and Storage to adjust data collection attributes for more even distribution of CPU load.
- Section 5.2.11, Display Oracle Statistics
- Section 5.2.12, Dump a Flat File
- Section 5.2.13, Bulk Data Export - (BDX) t to export data from numeric logs in History to a file that can then be directly imported by third-party applications such as spreadsheets.
- Section 5.2.14, Create/Drop User Indexes

The most commonly used functions have dedicated tabs as shown in Figure 5-1. These tabs provide easy access to interactive dialogs for database maintenance.

Functions that do not have a graphical user interface are accessed via the **Database Maintenance** tab. This tab has a **Maintenance Menu** button for displaying the on-line version of the database maintenance menu, Figure 5-2. This menu lets you access maintenance functions that are permitted with History running.



```
hsOutput.c      @ 395   | HS_LOG_ON is set, stdout & stderr logs are

<--------------Database Maintenance Program-------------->
<-------------------Local Node Options------------------->
<----------------History Running Options----------------->
<---NOTE: If the option you want is not listed, you may-->
<---------have to shut history down to see the option---->

 0.   Exit
 1.   Database Free Space Report
 2.   Entry Tables Report
 3.   Update Deadband Ratio Online.
 4.   Extend a Tablespace
 5.   Add/Alter/Delete/List NumericLog Directories
 6.   Reset the Object Status states
 7.   Cascade Attributes
 8.   Stagger Collection of data to improve performance
 9.   Display some oracle stats every 5 minutes
10.   Dump Flat File
11.   Bulk Data Export
12.   Create/Drop User Indexes


Enter Number> _
```

Figure 5-2. Database Maintenance Menu when History is Running

To invoke a function, enter the corresponding number, and then press the ENTER key. For example, to create or drop user indexes, enter **12** and then press the ENTER key. To exit the menu, choose **Exit**.

NOTE

- This menu can also be displayed by invoking hsDBMaint when History is running. See Section 5.2.2, hsDBMaint for Command Line Database Maintenance.
- Some database maintenance functions can also be executed via the DOS Command Prompt.

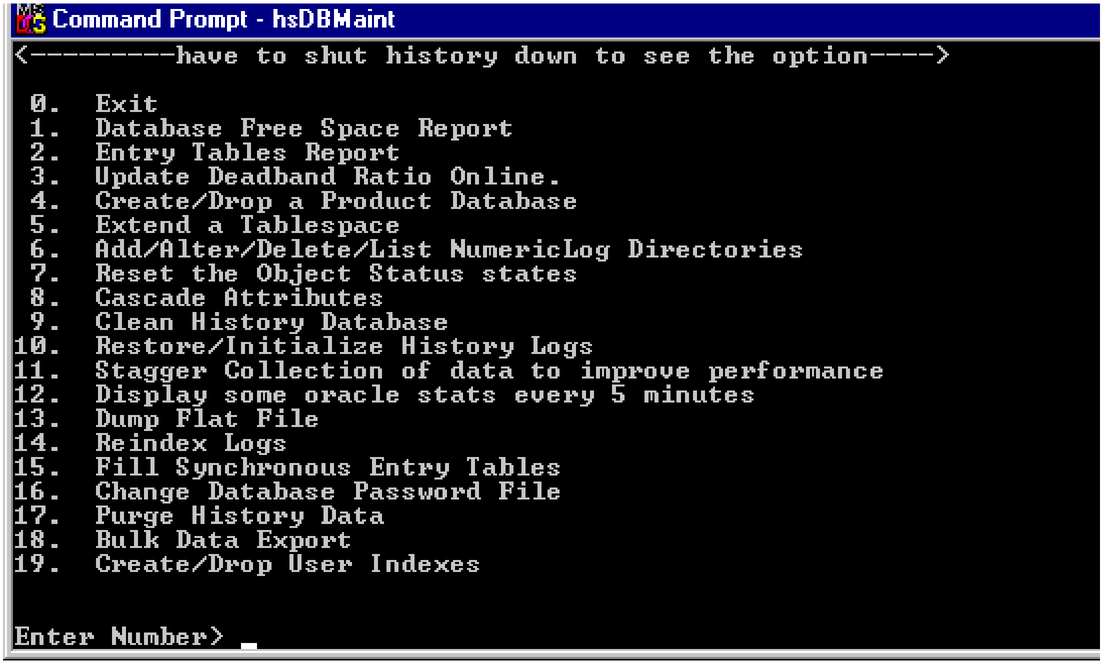
5.2.2 hsDBMaint for Command Line Database Maintenance

Certain database maintenance functions can not be accessed via the on-line History utilities. To access these database maintenance functions, invoke the hsDBMaint command from the DOS command prompt:

```
c:> “%HS_HOME%” \bin\hsDbMaint.exe
```

NOTE

Certain database maintenance functions MUST be performed with History shut down. The complete (off-line) menu of database maintenance functions is shown in Figure 5-3.



```
<---------have to shut history down to see the option---->

 0.   Exit
 1.   Database Free Space Report
 2.   Entry Tables Report
 3.   Update Deadband Ratio Online.
 4.   Create/Drop a Product Database
 5.   Extend a Tablespace
 6.   Add/Alter/Delete/List NumericLog Directories
 7.   Reset the Object Status states
 8.   Cascade Attributes
 9.   Clean History Database
10.   Restore/Initialize History Logs
11.   Stagger Collection of data to improve performance
12.   Display some oracle stats every 5 minutes
13.   Dump Flat File
14.   Reindex Logs
15.   Fill Synchronous Entry Tables
16.   Change Database Password File
17.   Purge History Data
18.   Bulk Data Export
19.   Create/Drop User Indexes


Enter Number> _
```

Figure 5-3. Database Maintenance Menu when History is off-line

To invoke a function, enter the corresponding number, and then press the ENTER key. For example, to clean the History database, enter **9** and then press the ENTER key. Then follow the instructions for the selected function. To exit the hsDBMaint function, choose **Exit** from the menu.

Refer to the following for details regarding off-line functions:

- Section 5.2.16, Create or Drop a Product Database
- Section 5.2.17, Clean History Database - As logs are created/deleted, if an operation is interrupted, it may leave dangling references in the database. For instance, if the creation of a log is not completed for some reason, the log name is used in the database even though the log does not actually exist. This means the log name can not be re-used. The cleanup function deletes these dangling references from the database. In the case of the dangling log name, this means that the name can be used again.
- Section 5.2.18, Restore or initialize History Logs
- Section 5.2.19, Reindex Logs
- Section 5.2.20, Fill Synchronous Entry Tables
- Section 5.2.21, Purge History Data - clear History data erroneously collected in the future.

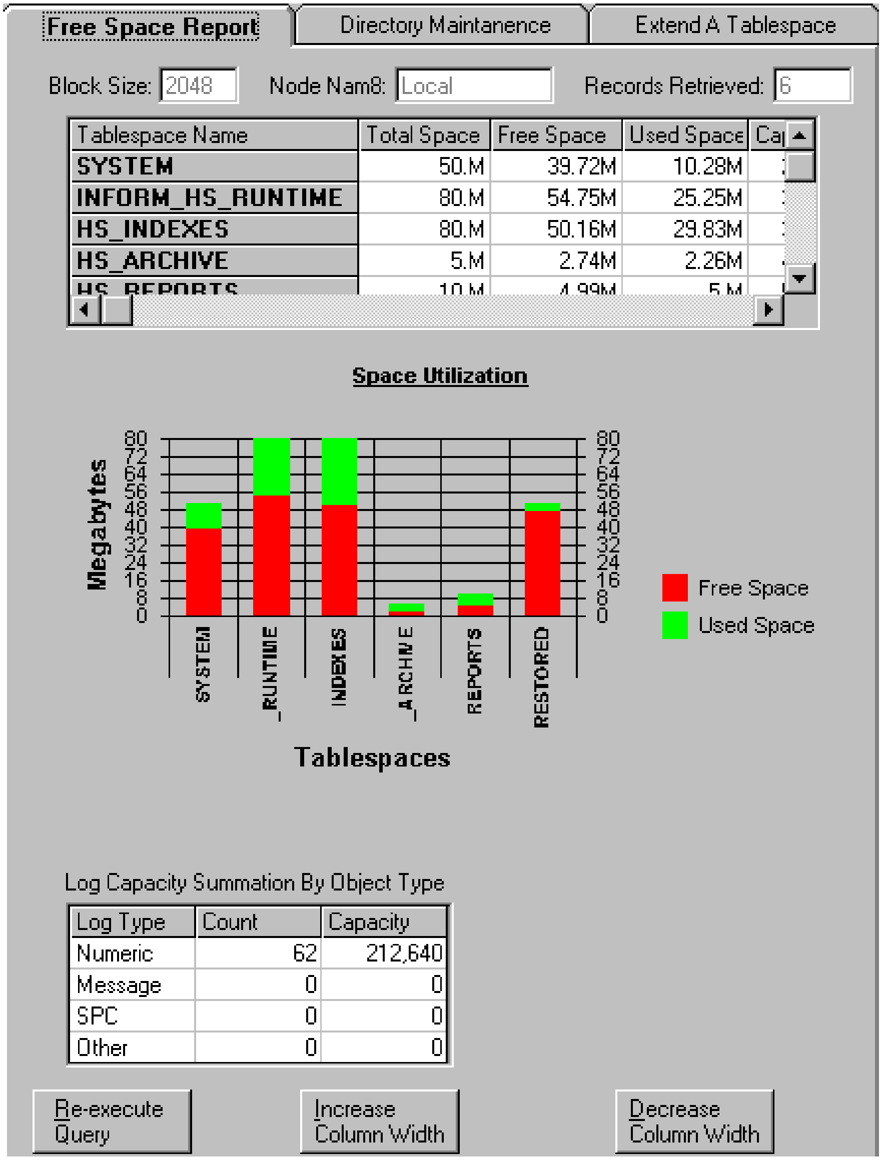
NOTE

- You can also invoke any of the on-line data base maintenance functions off-line via hsDBMaint.
- Some database maintenance functions can also be executed via the DOS Command Prompt.

5.2.3 Database Free Space Report

The Free Space Report indicates how much space is available in the tablespaces. To create a Free Space Report:

1. Choose **Tools > History Utilities** from the History Builder menu bar.
2. From the History Utilities, click the **Free Space Report** tab. This displays the free space report. An example is shown in Figure 5-4.
3. You can increase or decrease the width of any column in the Tablespace table (upper table) by clicking in the column and then clicking either **Increase Column Width** or **Decrease Column Width**.
4. Click the **Re-execute Query** button to update the report.



Block Size: 2048
Node Nam8: Local
Records Retrieved: 6

Tablespace Name	Total Space	Free Space	Used Space	Ca
SYSTEM	50.M	39.72M	10.28M	
INFORM_HS_RUNTIME	80.M	54.75M	25.25M	
HS_INDEXES	80.M	50.16M	29.83M	
HS_ARCHIVE	5.M	2.74M	2.26M	
HS_REPORTS	10.M	4.99M	5.M	

Log Capacity Summation By Object Type

Log Type	Count	Capacity
Numeric	62	212,640
Message	0	0
SPC	0	0
Other	0	0

Figure 5-4. Free Space Report GUI

5.2.4 Entry Tables Report

This function generates a report of all numeric logs, their current capacity, and other useful statistics. You can run this function from either the History Utility, or from the DOS command prompt.

If you run this function via the History Utility, you can modify these report settings before you run the report:

- log ID to process. You may choose to generate a report for all logs, just one log, or a range of logs. The default is for ALL LOGS.
- change the correction of bad logs. The default is NONE.
- change the line per page. The default is 24 lines per page.

If the above default settings are OK, it is best to run from the command line to allow output to be directed to a file. If you need to change a default setting, use the History Utility.

NOTE

This function may take a couple hours to complete for a large database, and generates a high CPU load.

5.2.4.1 Running Entry Tables Report from the Command Prompt

This generates a report for all logs and directs the report output to a specified file.

To run the report, from the DOS command prompt, enter:

```
c: > hsDBMaint -report > /tmp/filename.txt
```

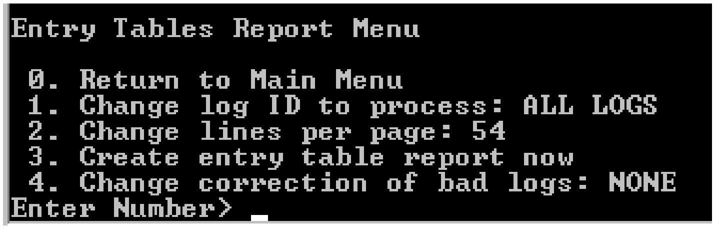
where *filename* is the name of the file where the report output will be directed to.

The file is given a .txt extension. You can view the file using an editor such as notepad.

5.2.4.2 Running Entry Tables Report from the History Utility

To run this function from the History Utility:

1. Choose **Tools > History Utilities** from the History Builder menu bar.
2. From the History Utility, click the **Database Maintenance** tab.
3. In the Database Maintenance tab, click the **Maintenance Menu** button. This displays the on-line version of the database maintenance menu, Figure 5-1.
4. Choose the **Entry Tables Report** option, and then press ENTER. This displays the Entry Tables Report Menu, Figure 5-5.



```
Entry Tables Report Menu

 0. Return to Main Menu
 1. Change log ID to process: ALL LOGS
 2. Change lines per page: 54
 3. Create entry table report now
 4. Change correction of bad logs: NONE
Enter Number> _
```

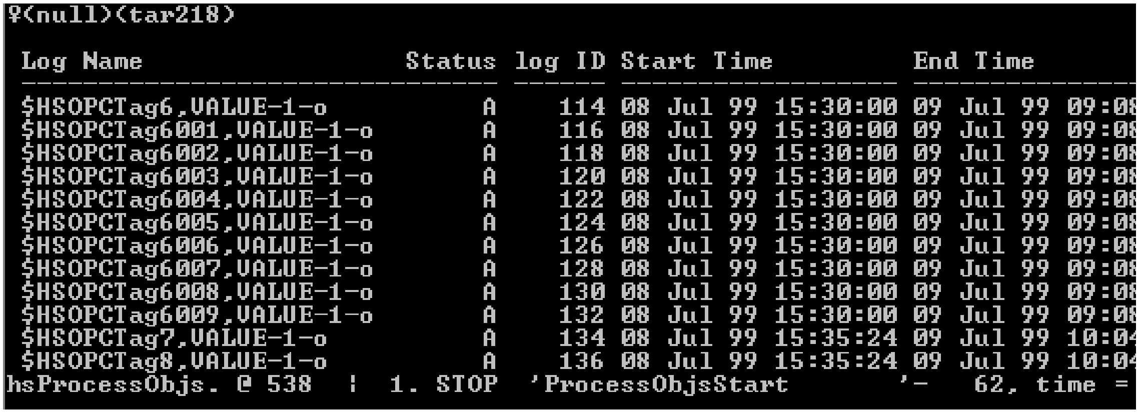
Figure 5-5. Entry Tables Report Menu

This menu shows you the current settings for log ID to process, correction of bad logs, and lines per page. You can change the current settings before you run the report, or you can return to the database maintenance menu without using any of these functions.

If you need to change any of the settings, choose the corresponding option from this menu and follow the dialog. When you finish with the dialog, you will return to the Entry Tables Report menu and the current settings will be changed according to the dialog. When the settings are correct, at the menu prompt, choose the **Create entry table report now** option, and then press ENTER. This generates the report in the Command Prompt window, Figure 5-6. A short summary is provided at the end of the report, Figure 5-7.

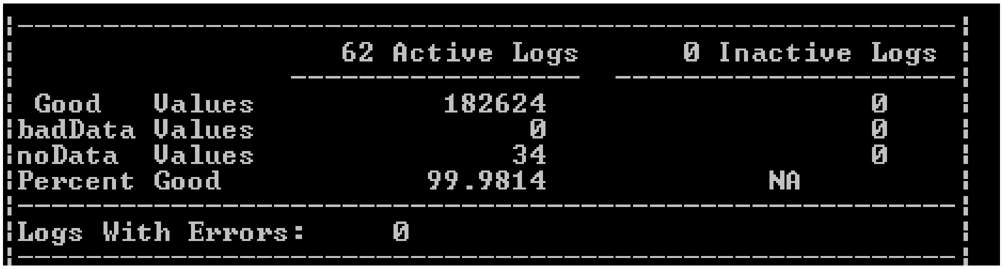
NOTE

You may have to adjust the height and width of the Command Prompt window to display the report properly. Use the **Properties** option in the window's right-click menu.



```
♀(null)(tar218)

 Log Name                         Status  log ID Start Time         End Time
 -------------------------------  ------  ------ ------------------ -------------
 $HSOPCTag6,VALUE-1-o                 A     114 08 Jul 99 15:30:00 09 Jul 99 09:0
 $HSOPCTag6001,VALUE-1-o              A     116 08 Jul 99 15:30:00 09 Jul 99 09:0
 $HSOPCTag6002,VALUE-1-o              A     118 08 Jul 99 15:30:00 09 Jul 99 09:0
 $HSOPCTag6003,VALUE-1-o              A     120 08 Jul 99 15:30:00 09 Jul 99 09:0
 $HSOPCTag6004,VALUE-1-o              A     122 08 Jul 99 15:30:00 09 Jul 99 09:0
 $HSOPCTag6005,VALUE-1-o              A     124 08 Jul 99 15:30:00 09 Jul 99 09:0
 $HSOPCTag6006,VALUE-1-o              A     126 08 Jul 99 15:30:00 09 Jul 99 09:0
 $HSOPCTag6007,VALUE-1-o              A     128 08 Jul 99 15:30:00 09 Jul 99 09:0
 $HSOPCTag6008,VALUE-1-o              A     130 08 Jul 99 15:30:00 09 Jul 99 09:0
 $HSOPCTag6009,VALUE-1-o              A     132 08 Jul 99 15:30:00 09 Jul 99 09:0
 $HSOPCTag7,VALUE-1-o                 A     134 08 Jul 99 15:35:24 09 Jul 99 10:0
 $HSOPCTag8,VALUE-1-o                 A     136 08 Jul 99 15:35:24 09 Jul 99 10:0
hsProcessObjs. @ 538  ¦  1. STOP  'ProcessObjsStart       '-   62, time =
```

Figure 5-6. Sample Entry Tables Report



```
¦---------------------------------------------------------¦
¦                   62 Active Logs      0 Inactive Logs   ¦
¦                ------------------  -------------------- ¦
¦ Good   Values           182624                     0    ¦
¦badData Values                0                     0    ¦
¦noData  Values               34                     0    ¦
¦Percent Good            99.9814                NA        ¦
¦---------------------------------------------------------¦
¦Logs With Errors:      0                                 ¦
¦---------------------------------------------------------¦
```

Figure 5-7. Report Summary

5.2.5 Update Deadband Ratio On-line

Use this function to update deadband ratio and estimated log time period for logs that use deadband.

Disk space is allocated to logs according to their log capacity, and not the amount of data actually stored. The log period can be adjusted on an individual log basis to keep log capacity at a reasonable size for each log. For all synchronous logs with or without data compaction, log capacity is calculated as follows:

> log capacity = (log period/storage interval)

When you use compaction, the log period is effectively increased so that you store the same number of samples (as determined by the log capacity) over a longer period of time. Consider the following two cases. If your process is very unstable such that every sample is stored and there is no data compaction, the effective log period will be equal to the configured log period. If your process is perfectly stable (maximum data compaction, samples stored only at deadband storage interval), the effective log period is extended according to the following formula:

> effective log period = (deadband storage interval/storage interval) * configured log period

In most cases, since the amount of process variation will not be at either extreme, the effective log period will fall somewhere in between the configured (minimum) and maximum value.

The effective log period calculated according to the above formula is stored in the EST_LOG_TIME_PER attribute. This attribute is updated periodically according to a schedule which repeats as often as every five days. If necessary, you can use the History Utility to manually update this attribute.

EST_LOG_TIME_PER is used by the Log Manager when it needs to determine which log to retrieve data from (seamless retrieval). For instance, consider an application with a composite log where the primary log stores data at 30 second intervals over a one-day time period, and the secondary log stores data at one-hour intervals over a one-week time period. Your data compaction ratio is 5:1 which extends the effective log period of the primary log to five days. If you make a request to retrieve data that is three days old, the log manager will know to get the data from the primary log rather than the secondary log, even through the configured log period of the primary log is only one day.

When EST_LOG_TIME_PER is updated, both internal History memory and the History database are updated with the new values of EST_LOG_TIME_PER and COMPACTION_RATIO. This will allow Log Managers access to the information and when the History node is restarted the information will also be available.

To update deadband ratio on-line:

1. Choose **Tools > History Utilities** from the History Builder menu bar.
2. From the History Utility, click the **Database Maintenance** tab.
3. In the Database Maintenance tab, click the **Maintenance Menu** button. This displays the on-line version of the database maintenance menu, Figure 5-1.
4. From the menu prompt, choose the **Update Deadband Ratio On-line** option, and then press ENTER.

NOTE

For a large database, this function may take several hours to complete.

5. At the prompt, enter **y** (Yes) to continue, or **n** (No) to cancel.

5.2.6 Extend Tablespace for History Files

Extend the tablespace in your system when you require greater storage capacity than is provided by the default tablespace sizes.

First refer to the Guidelines for Extending Tablespace. Then use the Extending Tablespace Procedure.

5.2.6.1 Guidelines for Extending Tablespace

Follow these guidelines to determine which tablespace(s) to extend and how much to extend them by:

- Determine how much more space you will require for additional log entries.

  Each Oracle-based numeric log entry occupies 45 bytes in the Inform_HS_RunTime tablespace. Multiply the number of additional entries by 45. This gives the additional space you need in bytes. For instance, if you need 100,000 additional log entries, then you must extend the Inform_HS_RunTime tablespace by 4.5 meg (100,000 * 45 = 4,500,000 bytes = 4.5 meg).

- When you extend the Inform_HS_RunTime tablespace, you must also extend the HS_Indexes tablespace by 80% of the amount that you extend the Inform_HS_RunTime tablespace by.

  For instance, if you add 4.5 meg to Inform_HS_RunTime tablespace, you must extend the Inform_HS_Indexes tablespace by 3.375 meg (4.5 meg * 80%).

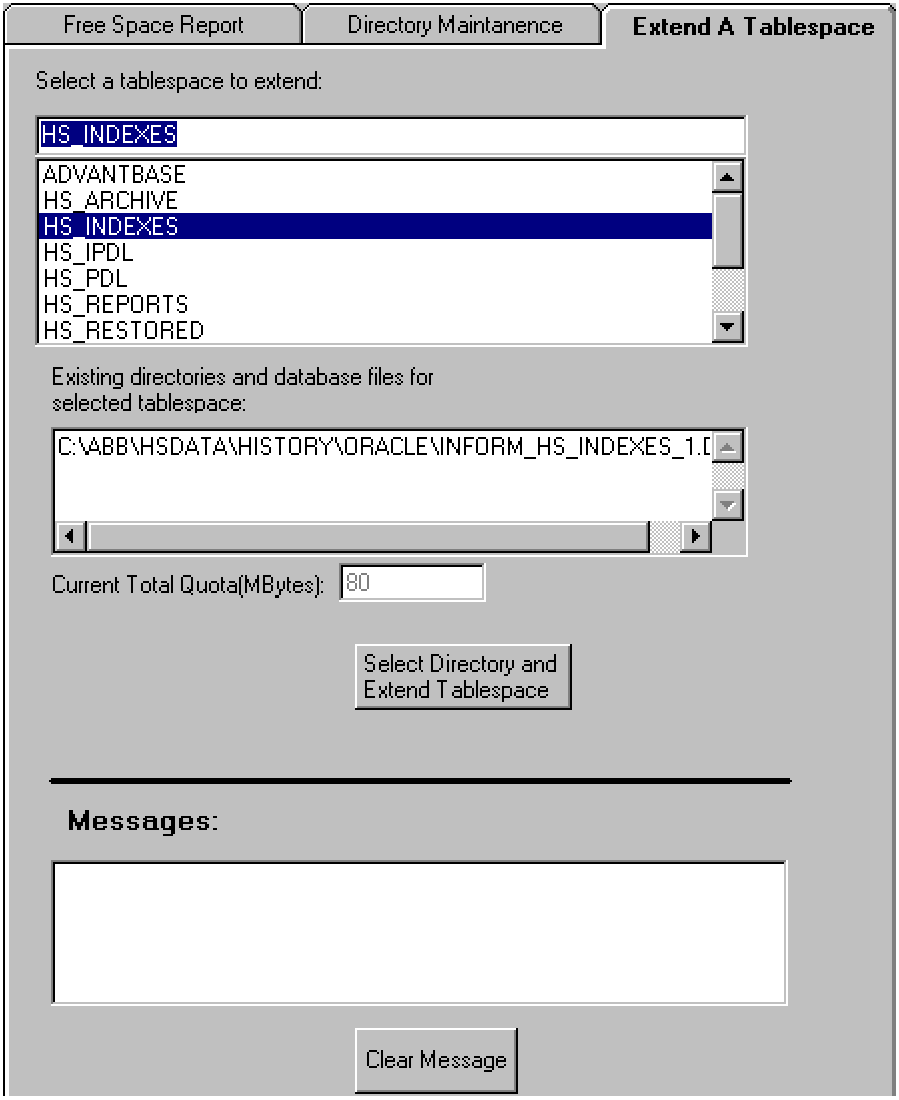
- Generally, you should not have to extend the HS_Restored tablespace.

  If you need to extend the HS_Restored tablespace, use the method above to determine how much space you will require (45 bytes per log entry), and extend the HS_Indexes tablespace by 80% of the amount that you extend the HS_Restored tablespace by.

5.2.6.2 Extending Tablespace Procedure

To extend a tablespace:

1. Choose **Tools > History Utilities** from the History Builder menu bar.
2. From the History Utility, click the **Extend A Tablespace** tab. This displays the dialog for selecting which tablespace to extend, Figure 5-8.
3. Select the tablespace to extend. This displays the following information for the selected tablespace: existing directories, database files, current total quota.



Select a tablespace to extend:

HS_INDEXES

ADVANTBASE
HS_ARCHIVE
HS_INDEXES
HS_IPDL
HS_PDL
HS_REPORTS
HS_RESTORED

Existing directories and database files for selected tablespace:

C:\ABB\HSDATA\HISTORY\ORACLE\INFORM_HS_INDEXES_1.D

Current Total Quota(MBytes): 80

Figure 5-8. Extend A Tablespace Tab

4. Click on **Select Directory and Extend Tablespace**. This displays a dialog for selecting the directory to write the new tablespace data to, and for specifying the file size, Figure 5-9.

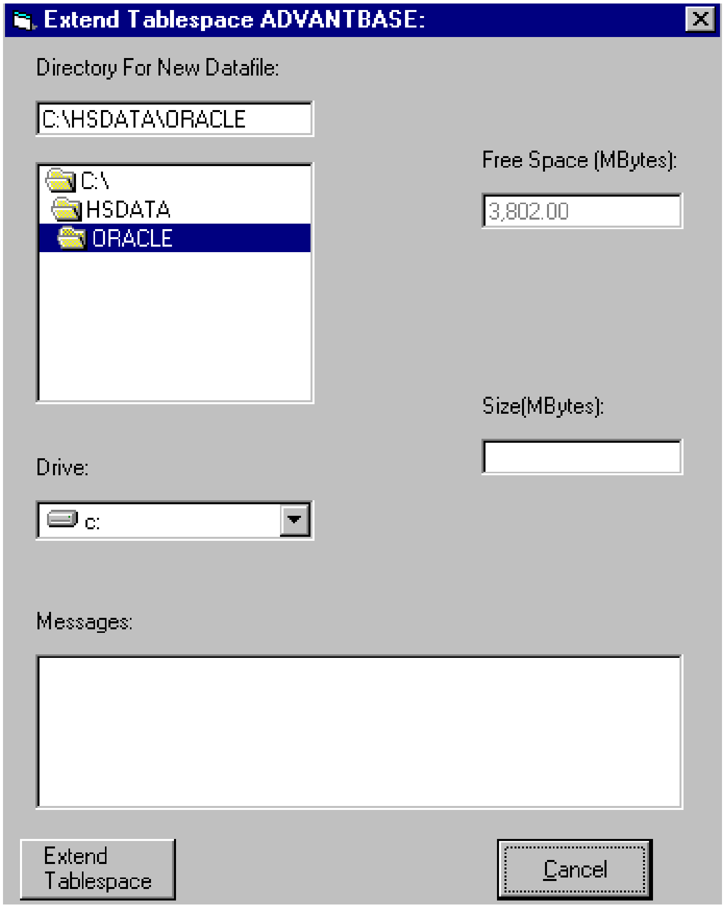


Figure 5-9. Extend Tablespace Dialog

5. Use the directory navigation window to select the directory for the data file. The free space for the selected directory is indicated in the corresponding field.

6. Enter the amount of this free space to allocate to the data file. This amount can not exceed the total free space. Click **Extend Tablespace** when you are done.

5.2.7 Maintain Directories for File-based Logs

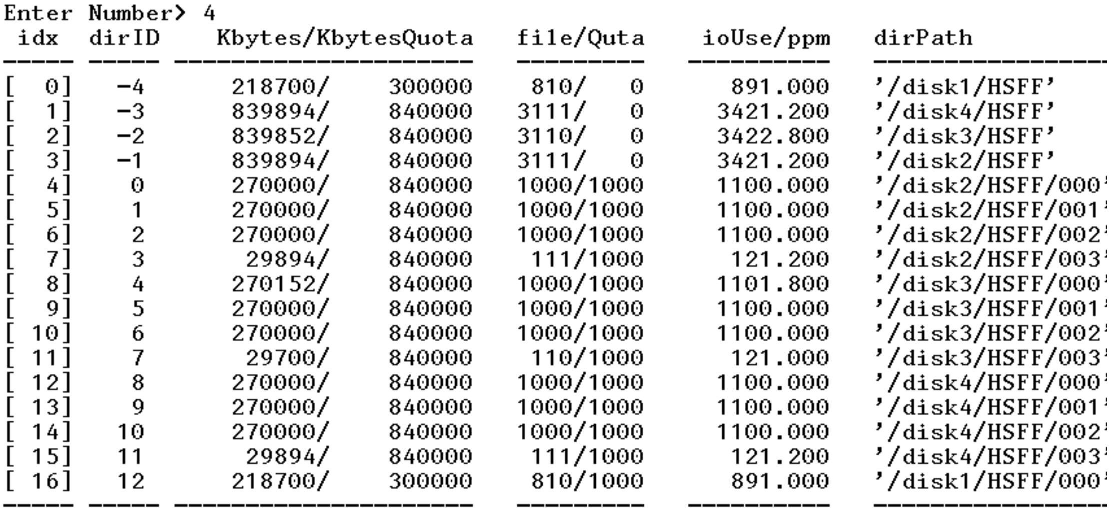
This function is used to configure and maintain a directory structure for file-based storage of numeric logs. File-based storage is applicable for numeric logs whose Storage Type is configured to be TYPE1 or TYPE2.

5.2.7.1 Overview

When History is installed, the disk that History resides on is allocated 200 MB for filed-based storage. You can change this allocation or allocate space on other disks via the History Utility.

To allocate space on a disk, you specify the path for the disk, and specify how much memory you want to allocate for files. The system automatically creates directories on the specified disks as needed.

The example directory structure shown in Figure 5-10 illustrates how file-based storage works. This example was produced using the List NumericLog Directories option of the off-line directory maintenance function (Section 5.2.2, hsDBMaint for Command Line Database Maintenance).



```
Enter Number> 4
 idx  dirID    Kbytes/KbytesQuota    file/Quta    ioUse/ppm    dirPath
----- ----- ---------------------   ---------   ----------   -----------------
[  0]   -4      218700/     300000    810/   0      891.000   '/disk1/HSFF'
[  1]   -3      839894/     840000   3111/   0     3421.200   '/disk4/HSFF'
[  2]   -2      839852/     840000   3110/   0     3422.800   '/disk3/HSFF'
[  3]   -1      839894/     840000   3111/   0     3421.200   '/disk2/HSFF'
[  4]    0      270000/     840000   1000/1000     1100.000   '/disk2/HSFF/000
[  5]    1      270000/     840000   1000/1000     1100.000   '/disk2/HSFF/001
[  6]    2      270000/     840000   1000/1000     1100.000   '/disk2/HSFF/002
[  7]    3       29894/     840000    111/1000      121.200   '/disk2/HSFF/003
[  8]    4      270152/     840000   1000/1000     1101.800   '/disk3/HSFF/000
[  9]    5      270000/     840000   1000/1000     1100.000   '/disk3/HSFF/001
[ 10]    6      270000/     840000   1000/1000     1100.000   '/disk3/HSFF/002
[ 11]    7       29700/     840000    110/1000      121.000   '/disk3/HSFF/003
[ 12]    8      270000/     840000   1000/1000     1100.000   '/disk4/HSFF/000
[ 13]    9      270000/     840000   1000/1000     1100.000   '/disk4/HSFF/001
[ 14]   10      270000/     840000   1000/1000     1100.000   '/disk4/HSFF/002
[ 15]   11       29894/     840000    111/1000      121.200   '/disk4/HSFF/003
[ 16]   12      218700/     300000    810/1000      891.000   '/disk1/HSFF/000
----- ----- ---------------------   ---------   ----------   -----------------
```

Figure 5-10. Example Directory Structure

When you specify a path for a disk, the directory HSFF is automatically created on the disk. For instance, when the path /disk4 is specified, the directory /disk4/HSFF is created. These HSFF files are assigned a negative dirID number which indicates the order in which their corresponding disks were added. For instance, in Figure 5-10, disk2 is assigned dirID number -1, disk3 has dirID number -2, and so on. The order in which disks are added determines the order they are used.

The HSFF directories have a file quota of 0 (zero). The files are actually stored in subdirectories of HSFF. These subdirectories are automatically created under the HSFF directory as files are stored.

The subdirectories are named using three-digit sequential numbers starting with 000. For instance, /disk4/HSFF/000. These subdirectories have a 1000-file limit (each log constitutes one file). When the number of files exceeds 1000, another subdirectory (/disk4/HSFF/001) is created to store the excess. Subdirectories are added as required until the quota on that disk is reached. New logs create files on the next disk. Disks are used in the order they are added.

In the example above, four disks have been specified for file-based storage: disk1, disk2, disk3, and disk4. The first disk added is disk2 (as indicated by dirID -1). A total of 3111 files are stored on disk in four directories. The directories disk2/HSFF/000, /001, and /002 each hold 1000 files. The directory disk2/HSFF/003 holds 111 and has the capacity for an additional 889 files. The disks disk3 and disk4 have similar patterns. The 003 directories do not contain more files because the total space usage would exceed the quota.

For further guidelines on directory maintenance, refer to Section 3.2.9, File Storage.

5.2.7.2 Directory Maintenance Procedures

Accessing the Directory Maintenance Utility

To access the directory maintenance utility:

1. Choose **Tools > History Utilities** from the History Builder main menu.
2. From the History Utility, click the **Directory Maintenance** tab. This displays the Directory Maintenance dialog, Figure 5-11.

This dialog lets you:

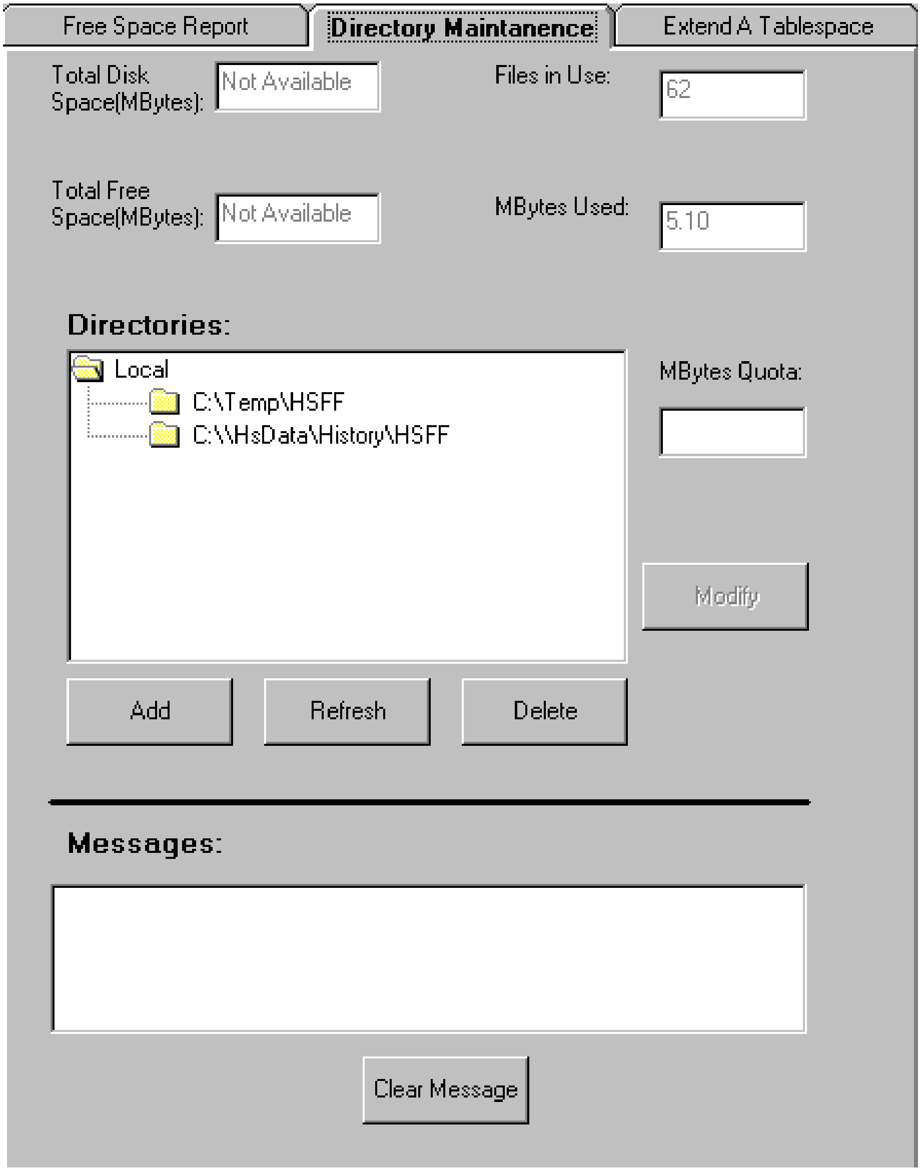
- View the Properties for a Selected Directory
- Modify the Memory Allocated to a Directory
- Delete a Directory from the List of Available Directories
- Add a Directory

You can also List Available Directories, but not via this dialog. If you need to display a list of existing directories, use the **Add, Alter, Delete, or List** option in the database maintenance menu (accessible via **Database Maintenance** tab).

View the Properties for a Selected Directory

To view the properties for a directory, simply select the directory from the list of available directories. The following information is provided:

- **Total Disk Space (MBytes)** - indicates available space on the selected drive.
- **Total Free Space (MBytes)** - indicates available space on the selected directory.
- **Files in Use** - indicates the number of files (logs) in the selected directory. The maximum number of files allowed in a directory is 1000.
- **Mbytes Used** - indicates the amount of space currently being used.
- **Mbytes Quota** - indicates the amount of memory allocated to the selected directory.



Total Disk Space(MBytes): Not Available

Files in Use: 62

Total Free Space(MBytes): Not Available

MBytes Used: 5.10

MBytes Quota:

Local
- C:\Temp\HSFF
- C:\\HsData\History\HSFF

Figure 5-11. Directory Maintenance Tab

Modify the Memory Allocated to a Directory

You can modify the Mbytes quota for any directory directly under the root directory, but not any of its sub-directories. To modify the quota for an existing directory, select the directory in the **Directories:** list. This activates the **Modify** button. Enter a new value in the MBytes Quota field, and then click **Modify**.

Delete a Directory from the List of Available Directories

To delete a directory, select the directory in the **Directories:** list, then click **Delete**.

NOTE

Delete is not allowed if there are files in the directory.

Add a Directory

To add a directory:

1. Click **Add**. This displays the Add Directory dialog, Figure 5-12.
2. Use the **Drive** pull down menu to select the drive where the directory is located, and then use the Root Directory window to navigate the directory structure and select a new directory for storing file-based logs. Make sure the disk where the directory is being added has enough space for the directory.

   When you make a selection, the available free space in the directory is indicated.
3. Enter the quota that you want to allocate to file storage in the **Quota** field. Enter the number in Megabytes. Typically, the disks you use for History storage are dedicated and not used for any other purpose. Therefore you can generally allocate the full disk capacity for file storage.
4. Click **Add New Root Directory** when you are finished.

A directory with the name HSFF is added under the path that you enter. For example, if you specify the path as `/disk3`, a new directory will be created as follows:

```
/disk3/HSFF
```

The first 1000 files (logs) will be stored in /HSFF/000.

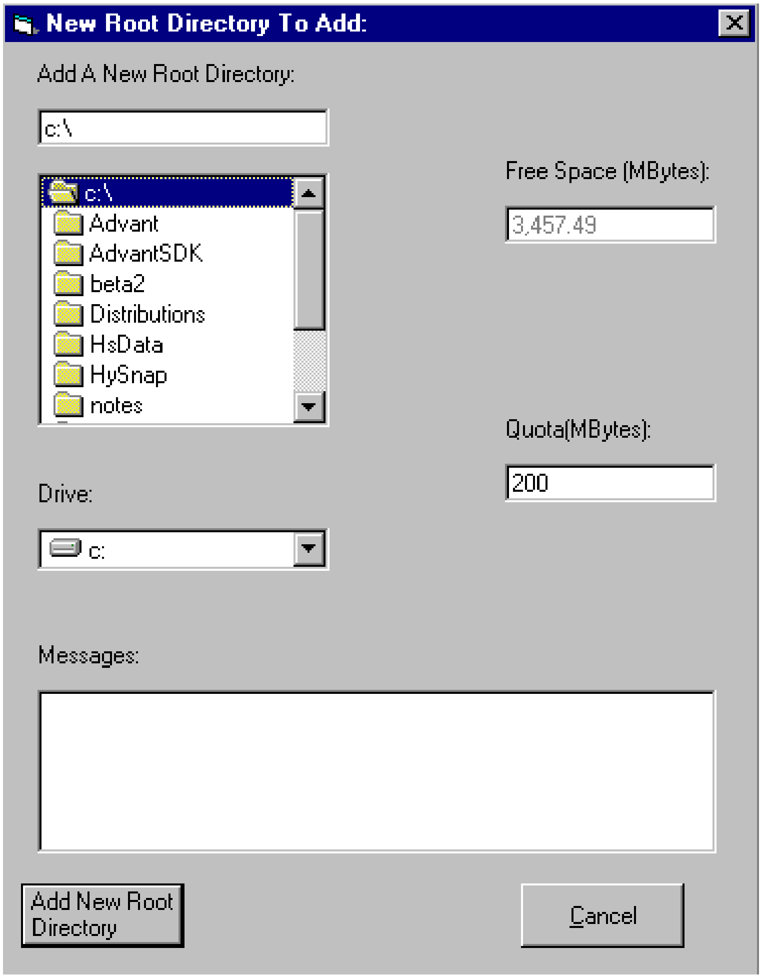
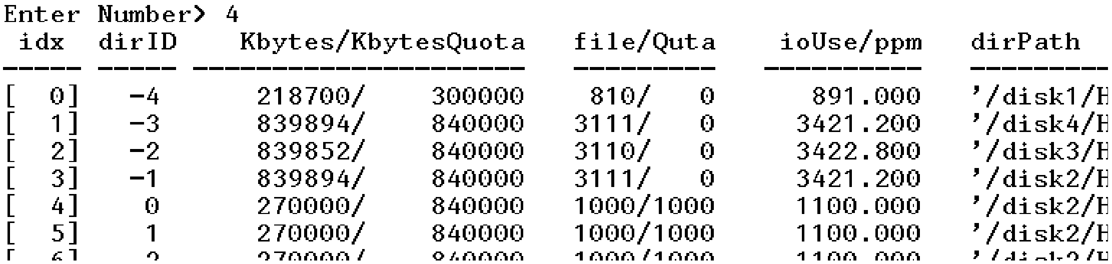


Figure 5-12. Add Directory Dialog

List Available Directories

To display a list of existing directories:

1. From the History Utility, click the **Database Maintenance** tab.
2. In the Database Maintenance tab, click the **Maintenance Menu** button. This displays the on-line version of the database maintenance menu, Figure 5-1.
3. From the menu prompt, choose the **Add/Alter/Delete/List NumericLog Directories** option, and then press ENTER. An example listing is shown in Figure 5-13.



```
Enter Number> 4
 idx  dirID    Kbytes/KbytesQuota    file/Quta    ioUse/ppm   dirPath
----- -----  ---------------------   ---------   ----------   ---------
[  0]   -4      218700/    300000     810/   0      891.000   '/disk1/H
[  1]   -3      839894/    840000    3111/   0     3421.200   '/disk4/H
[  2]   -2      839852/    840000    3110/   0     3422.800   '/disk3/H
[  3]   -1      839894/    840000    3111/   0     3421.200   '/disk2/H
[  4]    0      270000/    840000    1000/1000     1100.000   '/disk2/H
[  5]    1      270000/    840000    1000/1000     1100.000   '/disk2/H
```

Figure 5-13. Example Directory List

The columns in this list are as follows:

idx	Sequential index number
dirID	HSFF-level directories created under the disks that you add are assigned negative numbers, with -1 indicating the first disk added, and so on. Subdirectories under the HSFF-level directories are assigned positive numbers in the order they are created.
Kbytes	Indicates Kbytes used. For an HSFF-level directory, this is total usage for all subdirectories under it.
KbytesQuota	Indicates quota allocated for the directory. For an HSFF-level directory, this is total quota for all subdirectories under it.
file	Indicates number of files in the directory. For an HSFF-level directory, this represents the total number of files in all subdirectories under the HSFF directory.

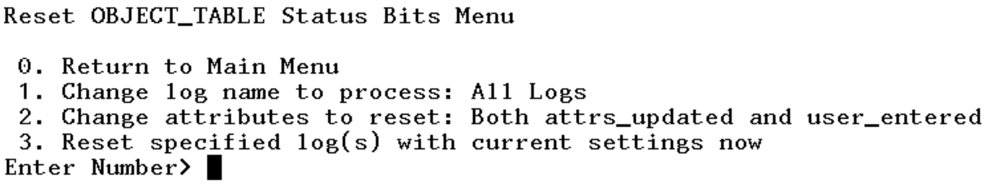
Quota	Indicates file quota. The File quota for HSFF directories is always zero (0) since files always stored in subdirectories. For subdirectories, the quota is always 1000.
ioUse/ppm	Average storage rate expressed in points per minute.
DirPath	This is the directory path.

5.2.8 Reset Object Status States

Use this function to reset *attrsUpdated* and *userEntered* bits. These bits determine whether the collection function for a log will update the data presentation attributes for numeric logs, or whether user-entered values will be used. If the *attrsUpdated* bit for a numeric log is set (1), when the log is activated, the collection function gets the data presentation attribute values from the data source. Any user-defined values will be overwritten. When the *userEntered* bit is set, it prevents the collection function from overwriting the user-defined values. At any given time, only one of these bits can be set for a numeric log. The Reset Object Status States function toggles the state for one or both bits.

To reset object status states:

1. Choose **Tools > History Utilities** from the History Builder main menu.
2. From the History Utility, click the **Database Maintenance** tab.
3. In the Database Maintenance tab, click the **Maintenance Menu** button. This displays the on-line version of the database maintenance menu, Figure 5-1.
4. From the menu prompt, choose the **Reset the Object Status states** option, and then press ENTER. This displays the reset OBJECT_TABLE Status Bits menu, Figure 5-14.



```
Reset OBJECT_TABLE Status Bits Menu

 0. Return to Main Menu
 1. Change log name to process: All Logs
 2. Change attributes to reset: Both attrs_updated and user_entered
 3. Reset specified log(s) with current settings now
Enter Number> █
```

Figure 5-14. Reset OBJECT_TABLE Status Bits Menu

This menu shows you the current settings for logs to process, and attributes to reset. You can change the current settings before you execute the reset operation, or return to the database maintenance menu without performing any of these functions.

If you need to change any of the settings, choose the corresponding option from this menu and follow the dialog. For log name to process you can choose:

- all logs
- all logs in a specific composite log hierarchy
- just one specific log

For attributes to reset you can choose:

- both *attrsUpdated* and *userEntered* bits
- just *attrsUpdated*
- just *userEntered*

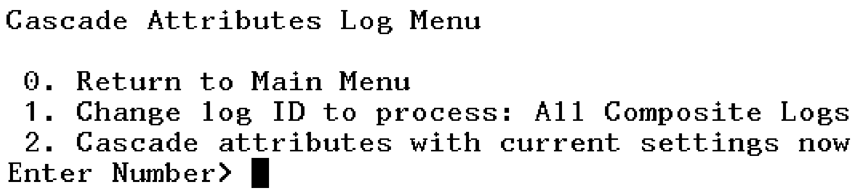
When the settings are correct, choose the **Reset specified log(s) with current settings now** option.

If you need to return to the database maintenance menu without invoking the Reset Object Status states function, choose the **Return to Main Menu** option.

5.2.9 Cascade Attributes for Composite Logs

This function propagates presentation, engineering units, and alarm attributes from the data source to secondary logs in the composite hierarchy (see Section 3.2.8.4, Alarm Attributes). This is required for logs where attribute values have been entered manually. To do this:

1. Choose **Tools > History Utilities** from the History Builder menu bar.
2. From the History Utility, click the **Database Maintenance** tab.
3. In the Database Maintenance tab, click the **Maintenance Menu** button. This displays the on-line version of the database maintenance menu, Figure 5-1.
4. From the menu prompt, choose the **Cascade Attributes** option, and then press ENTER. This displays the Cascade Attributes Log Menu, Figure 5-15.



```
Cascade Attributes Log Menu

 0. Return to Main Menu
 1. Change log ID to process: All Composite Logs
 2. Cascade attributes with current settings now
Enter Number> █
```

Figure 5-15. Cascade Attributes Log menu

5. Specify the log to pass attributes to. The default is for all logs as indicated in option 1. If this is acceptable, skip this step and go directly to step 3. To specify a specific log, enter **1** and then press the ENTER key. This displays the prompt: `Do you want to do all composite logs? [yn]`

   Enter **n**, and then press the ENTER key. This displays the following prompt:

   ```
   Enter the log ID of any log in the composite
   hierarchy>
   ```

   Enter the log name, and then press the ENTER key. This returns you to the Cascade Attributes Log Menu.

6. To cascade attributes with current setting for log, enter **2** and then press the ENTER key. This passes the applicable attributes from the data source to the specified log, and then returns you to the Cascade Attributes Log Menu.

5.2.10 Stagger Collection and Storage

Spikes may occur in the CPU load when sample and storage rates are configured such that a large number of messages are written to disk at any one time. The Blocking Rate and Activation Time log attributes can be used to adjust collection and storage of TTD data, and storage of primary OCS data for a more even distribution of CPU load. These log attributes may be defined initially when you configure the History database. If you do not configure these attributes initially, or if you find that adjustments are required, the Stagger Collection function provides a convenient means for adjusting these log attributes. This function must be run while History is running.

The Stagger Collection function provides two methods for distributing CPU load for collection and storage: Stagger and Phasing.

Stagger

Stagger is typically applied to a number of logs with the same alignment, storage interval, and sample interval, and where sample and storage interval are equal. Stagger assigns a blocking rate greater than the storage interval. The first Activation Time is modified to distribute the load. For example, consider an application with 120 logs with the following configuration:

- sample & storage interval = 30 seconds
- alignment = 12:00:00
- Blocking Rate = 150s

Without staggering, 120 messages with 5 entries each will be sent to storage every 2.5 minutes (150s). This will cause a spike in CPU usage. To distribute the load more evenly, you can use the Stagger Collection function to apply a blocking rate of 10m, and specify that 1/10 of the logs (12 messages with 20 entries each) be sent to storage every minute.

Phasing

Phasing is typically applied to a number of logs with the same alignment, storage interval, and sample interval, and the storage interval is significantly greater than the sample interval. For example, consider an application with 120 logs sampled at a 1-minute rate and an average calculated on a daily basis.

In this case, the 150s blocking rate would not prevent a CPU load spike at 00:00:00 (midnight). Phasing allows a set of logs to have slightly different blocking rates to prevent all logs from processing at the same time. For example, for a blocking rate of 30 minutes, phasing can be implemented by sub-dividing the 120 logs into groups and assigning different blocking rates to each group as follows:

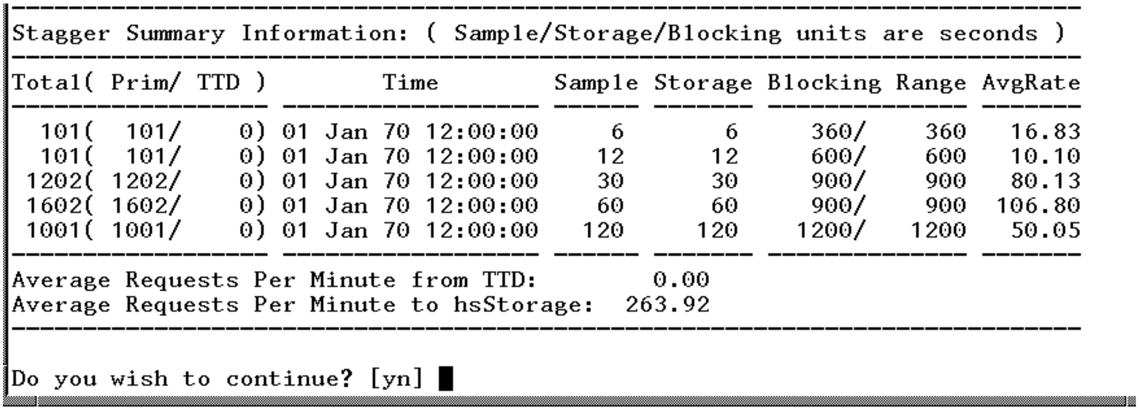
40 logs w/ Blocking Rate = 29m
40 logs w/ Blocking Rate = 30m
40 logs w/ Blocking Rate = 31m

In this case a spike will only occur once every 26,979 minutes (29x30x31).

How to Use the Stagger Collection Function

To run this function:

1. Choose **Tools > History Utilities** from the History Builder main menu.
2. From the History Utility, click the **Database Maintenance** tab.
3. In the Database Maintenance tab, click the **Maintenance Menu** button. This displays the on-line version of the database maintenance menu, Figure 5-1.
4. From the menu prompt, choose the **Stagger Collection of data to improve performance** option, and then press ENTER. This displays a summary of the current collection/storage performance, Figure 5-16.



```
---------------------------------------------------------------------------
Stagger Summary Information: ( Sample/Storage/Blocking units are seconds )
---------------------------------------------------------------------------
Total( Prim/ TTD )        Time        Sample Storage Blocking Range AvgRate
------------------ ------------------ ------ ------- -------------- -------
  101(  101/    0) 01 Jan 70 12:00:00     6       6     360/    360   16.83
  101(  101/    0) 01 Jan 70 12:00:00    12      12     600/    600   10.10
 1202( 1202/    0) 01 Jan 70 12:00:00    30      30     900/    900   80.13
 1602( 1602/    0) 01 Jan 70 12:00:00    60      60     900/    900  106.80
 1001( 1001/    0) 01 Jan 70 12:00:00   120     120    1200/   1200   50.05
------------------ ------------------ ------ ------- -------------- -------
Average Requests Per Minute from TTD:        0.00
Average Requests Per Minute to hsStorage:  263.92
---------------------------------------------------------------------------

Do you wish to continue? [yn] █
```

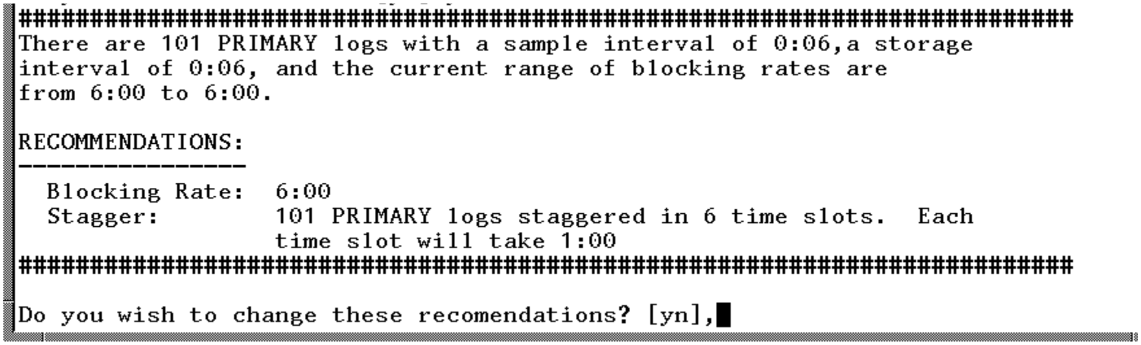
Figure 5-16. Summary of Collection/Storage Performance

The summary categorizes the logs by sample rate and storage rate. Each line of this summary provides the following information for a category of logs:

Total	Total number of logs in this category. This includes both cyclic subscriptions to OCS objects and periodic retrieval of TTD from controllers.
Prim	Number of logs in this category from cyclic subscriptions to OCS objects.
TTD	Number of logs in this category from periodic retrieval of TTD from controllers.
Time	First Activation time for logs in this category.
Sample	Sample rate for logs in this category (in seconds).
Storage	Storage rate for logs this category (in seconds).
Blocking Range	Shortest blocking rate for any logs in this category (in seconds)/ Longest blocking rate for any logs in this category (in seconds).
AvgRate	Calculated average number of requests per minute for this category based on number of logs, and sample, storage, and blocking rates. This equates to the average number of disk access transactions per minute for the system.

The prompt following the summary give you the opportunity to either exit History Utilities (by pressing `n`) or to continue with the Stagger function (by pressing `y`) .

If you choose to continue with the Stagger function, recommendations to improve performance for logs in the first category are displayed, Figure 5-17.



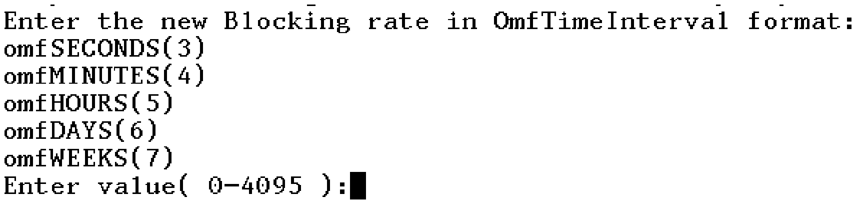
```
##########################################################################
There are 101 PRIMARY logs with a sample interval of 0:06,a storage
interval of 0:06, and the current range of blocking rates are
from 6:00 to 6:00.

RECOMMENDATIONS:
----------------
  Blocking Rate:  6:00
  Stagger:        101 PRIMARY logs staggered in 6 time slots.  Each
                  time slot will take 1:00
##########################################################################

Do you wish to change these recomendations? [yn],
```

Figure 5-17. Recommendations

The prompt following the recommendations gives you the opportunity to either accept the recommended blocking rate and stagger values for this category (by pressing `n`), or enter new values (by pressing `y`).

If you choose not to change the recommended values, you must respond to a prompt to confirm your selection, and then follow the same procedure for the next category (starting with recommendations for the next category).

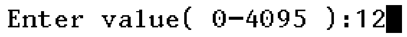
If you change the values, instructions for entering a new blocking rate are displayed, Figure 5-18.



```
Enter the new Blocking rate in OmfTimeInterval format:
omfSECONDS(3)
omfMINUTES(4)
omfHOURS(5)
omfDAYS(6)
omfWEEKS(7)
Enter value( 0-4095 ):
```

Figure 5-18. Instructions for Changing Blocking Rate

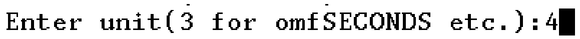
Enter a new blocking rate (without units) at the prompt. The range is 0 to 4095. For example, enter 12 as shown in Figure 5-19.



```
Enter value( 0-4095 ):12█
```

Figure 5-19. Example, Entering Blocking Rate

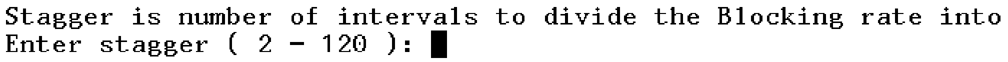
When the prompt for units is displayed, enter the units by entering the appropriate integer code: 3 = seconds, 4 = minutes, 5 = hours, 6 = days, 7 = weeks, Figure 5-20. In the example below, the integer code for minutes (4) is selected.



```
Enter unit(3 for omfSECONDS etc.):4█
```

Figure 5-20. Prompt for Units

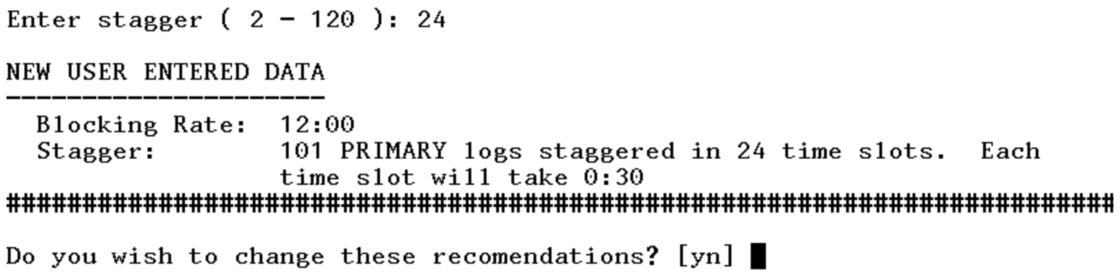
When the prompt for stagger is displayed, Figure 5-21, enter an integer value within the specified range. The range is based on the blocking rate that you specified.



```
Stagger is number of intervals to divide the Blocking rate into
Enter stagger ( 2 - 120 ): █
```

Figure 5-21. Prompt for Stagger

The stagger value specifies the number of divisions within the specified blocking rate that will be used to distribute the load. After you enter the stagger value, a summary of the changes you have made is displayed, Figure 5-22.

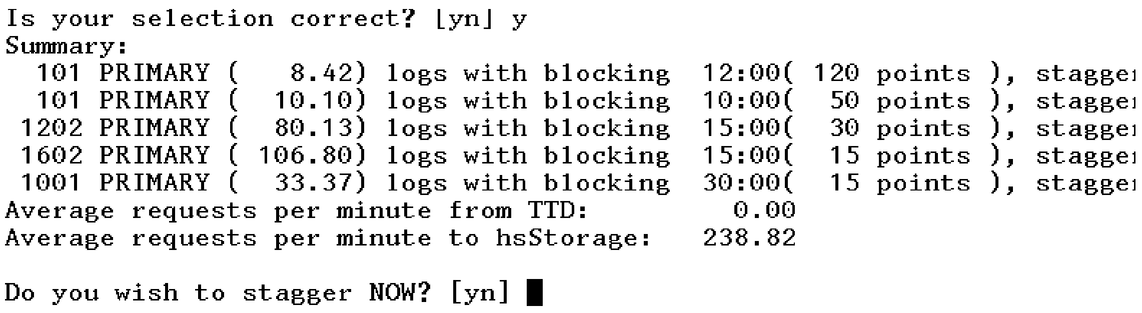


```
Enter stagger ( 2 - 120 ): 24

NEW USER ENTERED DATA
---------------------
  Blocking Rate:  12:00
  Stagger:        101 PRIMARY logs staggered in 24 time slots.  Each
                  time slot will take 0:30
##########################################################################

Do you wish to change these recomendations? [yn] █
```

Figure 5-22. Summary of Changes

The prompt following this summary gives you the opportunity to either accept these values (by pressing n), or change these values (by pressing y). In this case, the specified blocking rate is 12 minutes. The selected stagger divides each blocking interval into 24 30-second time slots. Thus at every 30-second time slot,

This procedure must be repeated for each category. When you have completed this for each category, a new summary will be displayed, and you will be prompted choose whether or not you want to apply the changes you have made, Figure 5-23.



```
Is your selection correct? [yn] y
Summary:
  101 PRIMARY (   8.42) logs with blocking  12:00( 120 points ), stagge
  101 PRIMARY (  10.10) logs with blocking  10:00(  50 points ), stagge
 1202 PRIMARY (  80.13) logs with blocking  15:00(  30 points ), stagge
 1602 PRIMARY ( 106.80) logs with blocking  15:00(  15 points ), stagge
 1001 PRIMARY (  33.37) logs with blocking  30:00(  15 points ), stagge
Average requests per minute from TTD:       0.00
Average requests per minute to hsStorage:   238.82

Do you wish to stagger NOW? [yn] █
```

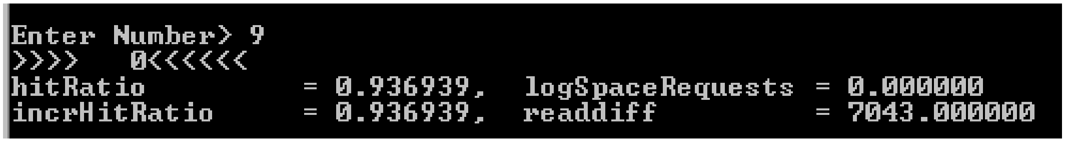
Figure 5-23. Revised Summary with Prompt to Apply Changes

Press `y` to apply the new stagger values, or `n` to ignore the changes you have made. The must be restarted in order for the stagger changes to take effect.

5.2.11 Display Oracle Statistics

Use this option to display Oracle statistics. The data are updated every five minutes. To do this:

1. Choose **Tools > History Utilities** from the History Builder main menu.
2. From the History Utility, click the **Database Maintenance** tab.
3. In the Database Maintenance tab, click the **Maintenance Menu** button. This displays the on-line version of the database maintenance menu, Figure 5-1.
4. From the menu prompt, choose the **Display some oracle stats every five minutes** option, and then press ENTER. An example is shown in Figure 5-24.



```
Enter Number> 9
>>>>   0<<<<<<
hitRatio          = 0.936939,  logSpaceRequests = 0.000000
incrHitRatio      = 0.936939,  readdiff         = 7043.000000
```

Figure 5-24. Example, Displaying Oracle Statistics

5.2.12 Dump a Flat File

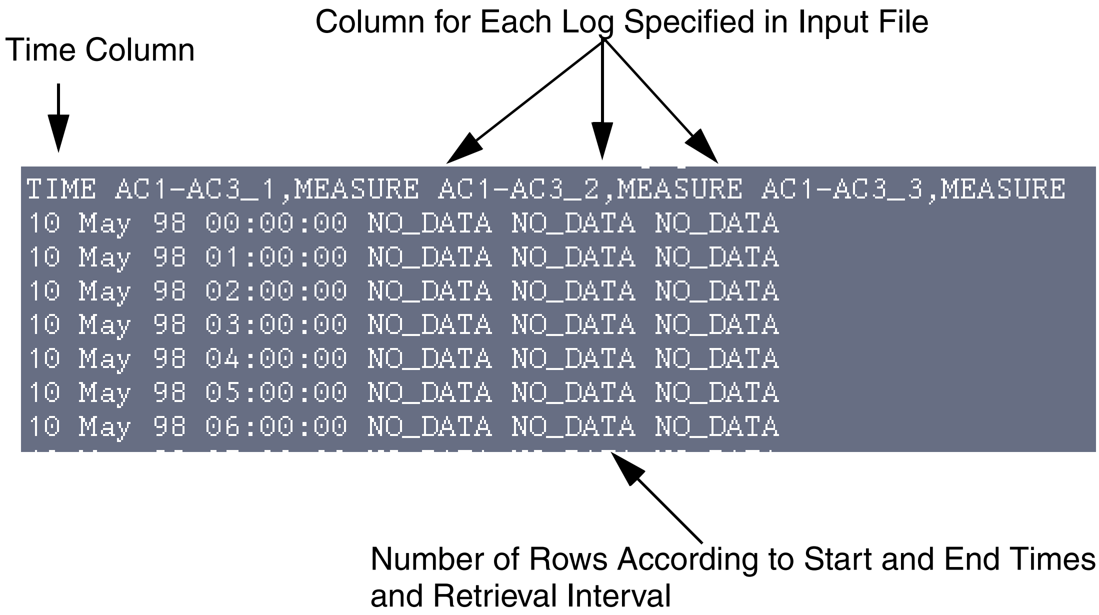
To run this function from the History Utility:

1. Choose **Tools > History Utilities** from the History Builder menu bar.
2. From the History Utility, click the **Database Maintenance** tab.
3. In the Database Maintenance tab, click the **Maintenance Menu** button. This displays the on-line version of the database maintenance menu, Figure 5-1.
4. From the Maintenance menu, choose **Dump a Flat File**.

5.2.13 Bulk Data Export

The Bulk Data Export (BDX) tool lets you export data from numeric logs in History to a file that can then be directly imported by third-party applications such as spreadsheets. The input to BDX is a text file where you specify a start time, end time, time interval between points, and log names. From this input file BDX produces an ASCII output file which contains a matrix of interpolated log values with corresponding log names and times, Figure 5-25.

This output file can be up to 100Meg (10,080 points x 500 logs max). You can use parameters in the input file to customize the output file format.


Time Column
Column for Each Log Specified in Input File
Number of Rows According to Start and End Times and Retrieval Interval

```
TIME AC1-AC3_1,MEASURE AC1-AC3_2,MEASURE AC1-AC3_3,MEASURE
10 May 98 00:00:00 NO_DATA NO_DATA NO_DATA
10 May 98 01:00:00 NO_DATA NO_DATA NO_DATA
10 May 98 02:00:00 NO_DATA NO_DATA NO_DATA
10 May 98 03:00:00 NO_DATA NO_DATA NO_DATA
10 May 98 04:00:00 NO_DATA NO_DATA NO_DATA
10 May 98 05:00:00 NO_DATA NO_DATA NO_DATA
10 May 98 06:00:00 NO_DATA NO_DATA NO_DATA
```

Figure 5-25. Example Output File

NOTE

If a large number of logs are specified, it may not be practical to view the output with a standard text editor due to the excessive width.

To run this function:

1. Choose **Tools > History Utilities** from the History Builder main menu.
2. From the History Utility, click the **Bulk Data Export** tab. This displays the Bulk Data Export dialog, Figure 5-26.

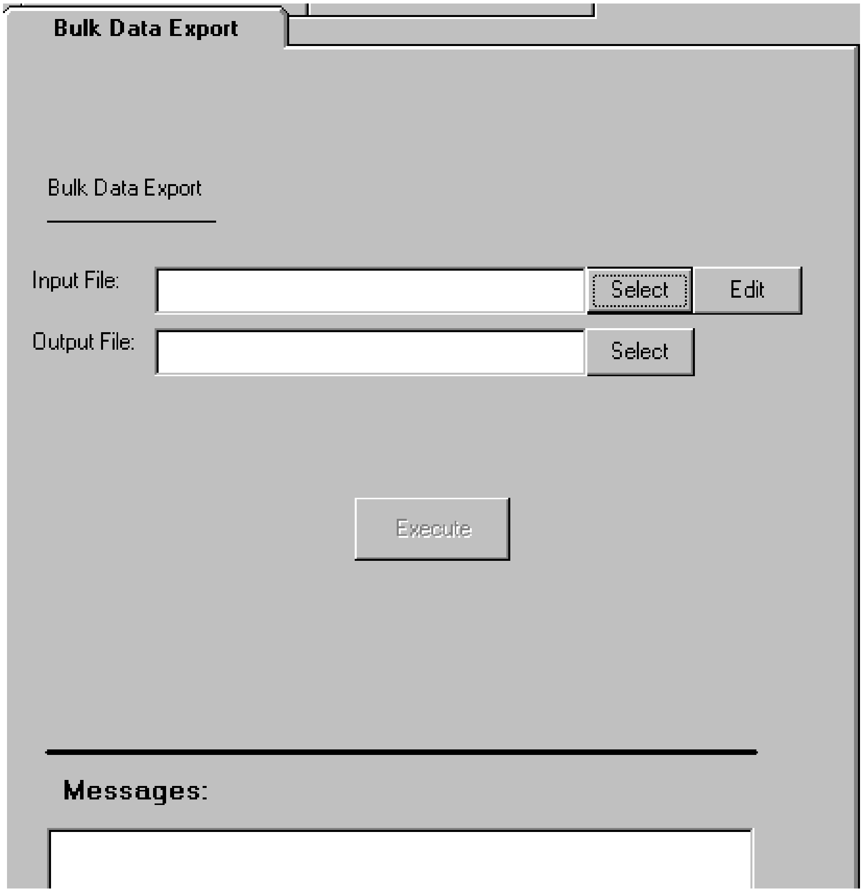


Figure 5-26. Dialog for Input/Output File Specification

3. Create the input text file. To do this:
   a. Click the Input File **Select** button.This displays a directory/file navigation dialog, Figure 5-27.

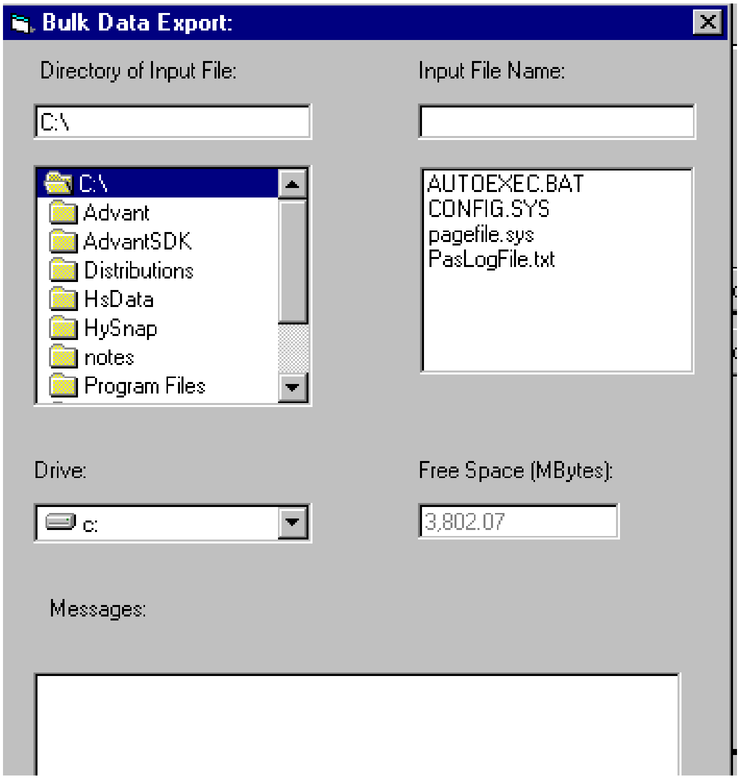
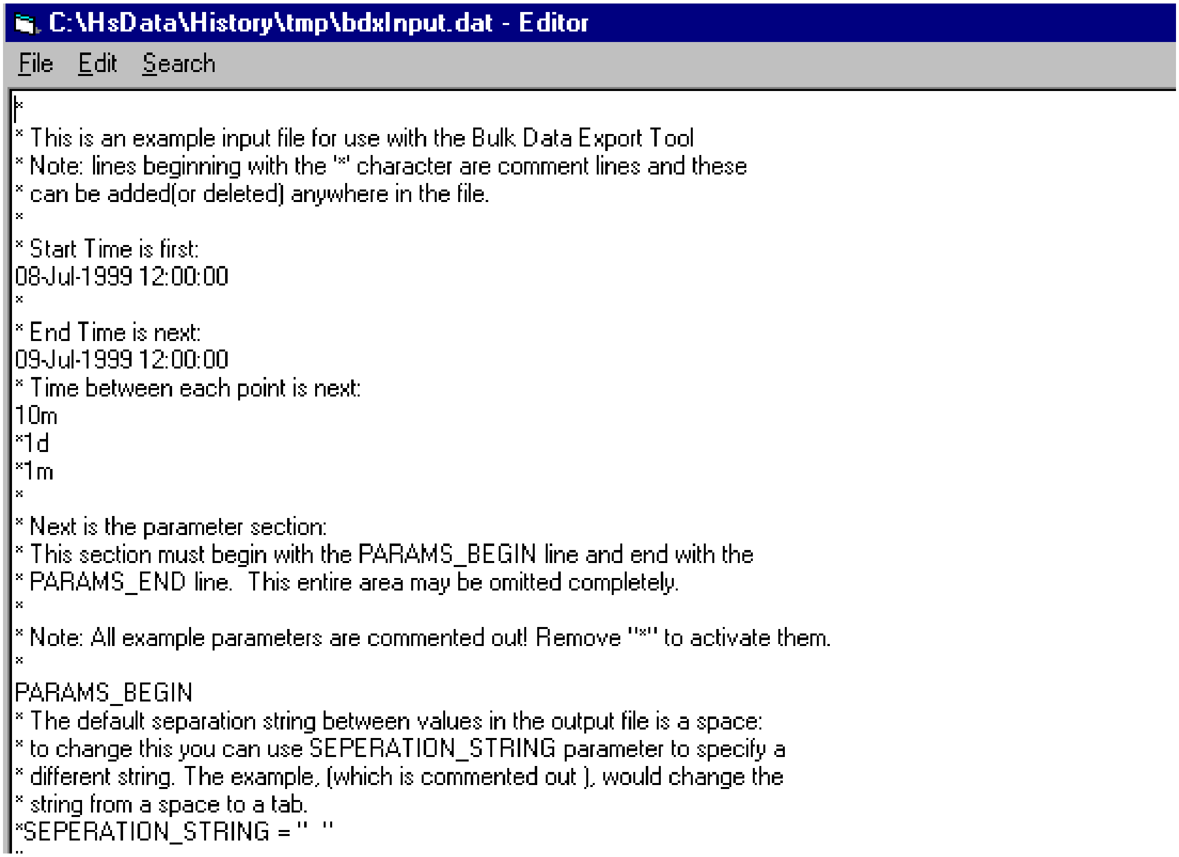


Figure 5-27. Bulk Data Export

b. Use this dialog to select an input file. A template file is provided to facilitate input file creation. This file named `bdxInput.dat` and is located in `c:\HsData\History\tmp`.

c. Click the **Edit** button to open a text editor, and edit the input file according to Section 5.2.13.1, Creating an Input File. An example file is shown in Figure 5-28.



```
*
* This is an example input file for use with the Bulk Data Export Tool
* Note: lines beginning with the '*' character are comment lines and these
* can be added(or deleted) anywhere in the file.
*
* Start Time is first:
08-Jul-1999 12:00:00
*
* End Time is next:
09-Jul-1999 12:00:00
* Time between each point is next:
10m
*1d
*1m
*
* Next is the parameter section:
* This section must begin with the PARAMS_BEGIN line and end with the
* PARAMS_END line.  This entire area may be omitted completely.
*
* Note: All example parameters are commented out! Remove "*" to activate them.
*
PARAMS_BEGIN
* The default separation string between values in the output file is a space:
* to change this you can use SEPERATION_STRING parameter to specify a
* different string. The example, (which is commented out ), would change the
* string from a space to a tab.
*SEPERATION_STRING = "  "
```

Figure 5-28. Example Input File for Bulk Data Export

4. Create the output file. To do this click the Output File Select button. This displays another directory/file navigation dialog similar to the one shown in Figure 5-27. Use this dialog to select an output file.

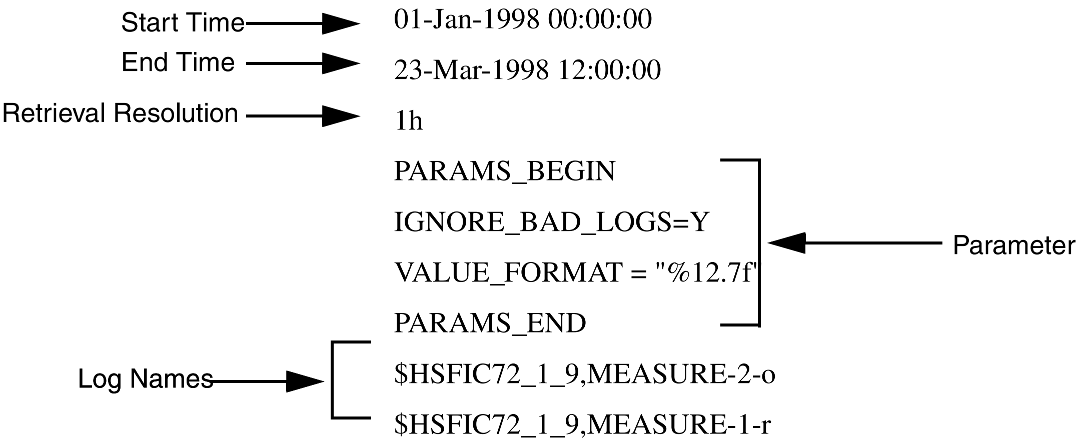
5. When you are finished with the input and output file specifications, click **Execute**.

NOTE

Rather than run the Bulk Data Export tool manually, you can write a C-language program to run the Bulk Data Export tool programmatically. An example program is provided with the Bulk Data Export tool. To open this file, choose **Get a C language example callbdx.c** (option 5) from the Bulk Data Export menu. See Running Bulk Data Export from the Command Prompt.

5.2.13.1 Creating an Input File

You must create a text file to specify contents and format of the Bulk Data Export output file. This text file has three parts. An example is shown in Figure 5-29.



```
01-Jan-1998 00:00:00
23-Mar-1998 12:00:00
1h
PARAMS_BEGIN
IGNORE_BAD_LOGS=Y
VALUE_FORMAT = "%12.7f"
PARAMS_END
$HSFIC72_1_9,MEASURE-2-o
$HSFIC72_1_9,MEASURE-1-r
```

Figure 5-29. Example Input File

NOTE

An example input file is provided with the Bulk Data Export tool and may be used as a starting point for creating your own input file. To open this file, choose **Get an example input file** (option 4) from the Bulk Data Export menu, Section 5.2.13.3, Running Bulk Data Export from the Command Prompt.

Start Time, End Time, Time Interval Between Points

This part is mandatory and determines the output file length (number of rows). Enter this information in the following order:

- Enter the request start time in ASCII format on the first line. For example: `01-Jan 1998 12:00:00`.
- Enter the end time in ASCII format on the second line. For example: `23-Mar 1998 12:00:00`.
- Enter the retrieval resolution on the third line. For example: `10m` or `1h` or `1d`, and so on.

Parameter List

The parameter list is optional. This is where you can specify parameters to customize the format of the output file. If you do not use parameters, the default formats are used. If you include a parameter list, it must start with a *begin* marker line: **`PARAMS_BEGIN`** and end with an ending marker line: **`PARAMS_END`**. The parameters are described in Section 5.2.13.2, Bulk Data Export Parameters.

Log Names

Log names (one minimum) are mandatory. They specify which logs are included in the output file. This also determines the output file width (number of columns). Enter one log per line. Logs can be either original, restored or composite.

There is a performance penalty for using composite logs as well as oracle-based logs, like all restored logs. When you use a composite log, the "best" original log is selected according to the selection criteria as described in Section 4.4, Criteria for Selection for Seamless Retrieval. A restored log is used if no original log fits the request.

5.2.13.2 Bulk Data Export Parameters

SEPARATION_STRING

The default separation string between values in the output file is a single space. To change this you can use SEPARATION_STRING parameter to specify a different string. This example changes the string to a tab:

SEPARATION_STRING = "	"

IGNORE_BAD_LOGS

By default, any log that can not be resolved causes the tool to exit with an error status. The parameter IGNORE_BAD_LOGS lets the tool continue if a log cannot be resolved. Check `hsBDX.err` file in the same directory as the output file for a list of errors/warnings.

Example:

IGNORE_BAD_LOGS=Y

NO_DATE

The date for each row of points is printed on the far left of each line by default. The dates can be deleted by setting NO_DATE to a true value. Example:

NO_DATE=y

NO HEADER

The first line of output file contains a blank separated set of valid logs, by default, as specified below. This line can be eliminated by the NO HEADER parameter s follows:

NO_HEADER=y

FIXED_FORMAT

By default values are printed using the "%g" variable length format, see printf man page for further details. The format can be changed to a fixed format, "%12.8g" by using the following boolean parameter:

FIXED_FORMAT = true

You cannot use FIXED_FORMAT in combination with VALUE_FORMAT. They are mutually exclusive.

VALUE_FORMAT

The value format can be specified using the parameter VALUE_FORMAT. Use only a single floating point specification, see the printf man page for details. Example:

VALUE_FORMAT = "%12.7f"

You cannot use VALUE_FORMAT in combination with FIXED_FORMAT. They are mutually exclusive.

TIME_FORMAT

The default time format can be overridden, see strftime man page. Example:

TIME_FORMAT = "%d-%B-%Y %T"

BAD_DATA_STRING

If a value has bad data quality the string BAD_DATA replaced a value in the table, by default. This string can be changed using the following parameter:

BAD_DATA_STRING ="BAD--DATA"

BAD_DATA_CHAR

Bad data quality can also be shown by appending a character to the received value by using the following parameter:

BAD_DATA_CHAR ="?"

The result would look like this ?50.00454 instead of BADDATA default.

NULL_DATA_STRING

If no data is found for a point at a given time the string NO_DATA is placed in the table instead of a value. This string can be changed using the following parameter:

```
NULL_DATA_STRING ="  --NO-DATA-"
```

5.2.13.3 Running Bulk Data Export from the Command Prompt

To invoke this function from the command line, enter:

hsDBMaint -bdx -i <*inputfilepath*> **-o** <*outputfilepath*>

For example:

```
hsDBMaint -bdx -i
-o
```

5.2.14 Create/Drop User Indexes

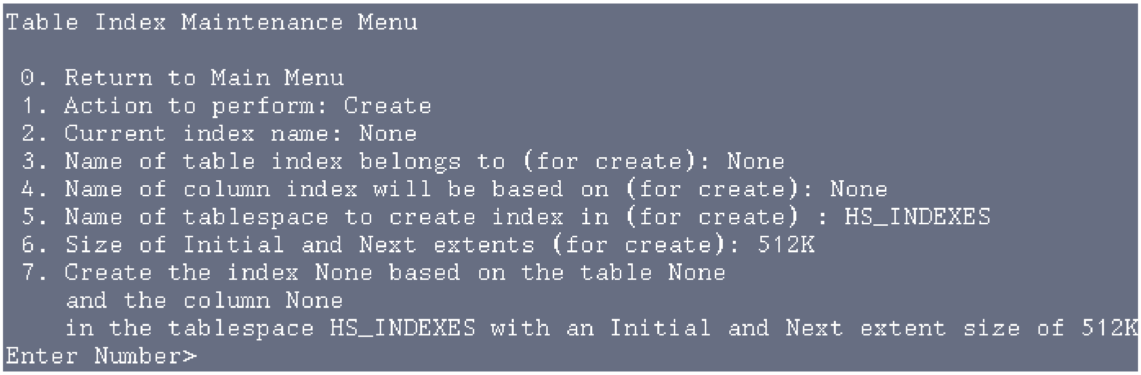
This function lets you create and drop user indexes to improve performance for SQL queries.

NOTE

History MUST be shut down prior to creating or dropping indexes.

To access this function:

1. Choose **Tools > History Utilities** from the History Builder main menu.
2. From the History Utility, click the **Database Maintenance** tab.
3. In the Database Maintenance tab, click the **Maintenance Menu** button. This displays the on-line version of the database maintenance menu, Figure 5-1.
4. From the menu prompt, choose the **Create/Drop User Indexes** option, and then press ENTER.This displays the Table Index Maintenance menu, Figure 5-30.



```
Table Index Maintenance Menu

 0. Return to Main Menu
 1. Action to perform: Create
 2. Current index name: None
 3. Name of table index belongs to (for create): None
 4. Name of column index will be based on (for create): None
 5. Name of tablespace to create index in (for create) : HS_INDEXES
 6. Size of Initial and Next extents (for create): 512K
 7. Create the index None based on the table None
    and the column None
    in the tablespace HS_INDEXES with an Initial and Next extent size of 512K
Enter Number>
```

Figure 5-30. Table Index Maintenance Menu

5.2.14.1 Dropping an Index

To drop an index:

1. Table Index Maintenance menu **Action to perform** (**1**), and then enter `d` to select the drop action.
2. Choose **Current index name** (**2**), and then enter the name of the index that you want to drop. Press <**Enter**> to display a list of index names.
3. Choose **Drop the index** (**7**) to drop the selected index.

5.2.14.2 Creating an Index

To create an index:

1. Table Index Maintenance menu, choose **Action to perform** (**1**), and then enter `c` to select the create action.
2. Choose **Current index name** (**2**), and then enter the name for the new index.
3. Choose **Name of table index belongs to** (**3**), and then enter the name of the table. Press <**Enter**> to display a list of table names.
4. Choose **Name of column index will be based on** (**4**), and then enter the column names. Separate each name with a comma, and DO NOT enter any white space between names. Press <**Enter**> to display a list of column names.
5. Choose **Name of tablespace to create index in** (**5**), and then enter the tablespace name. Press <**Enter**> to display a list of tablespace names.
6. Choose **Size of Initial and Next extent(s)** (**6**), and then enter the size, in kilobytes.
7. Choose **Create the index...** (**7**) to create the new index based on your entries for steps 3-6.

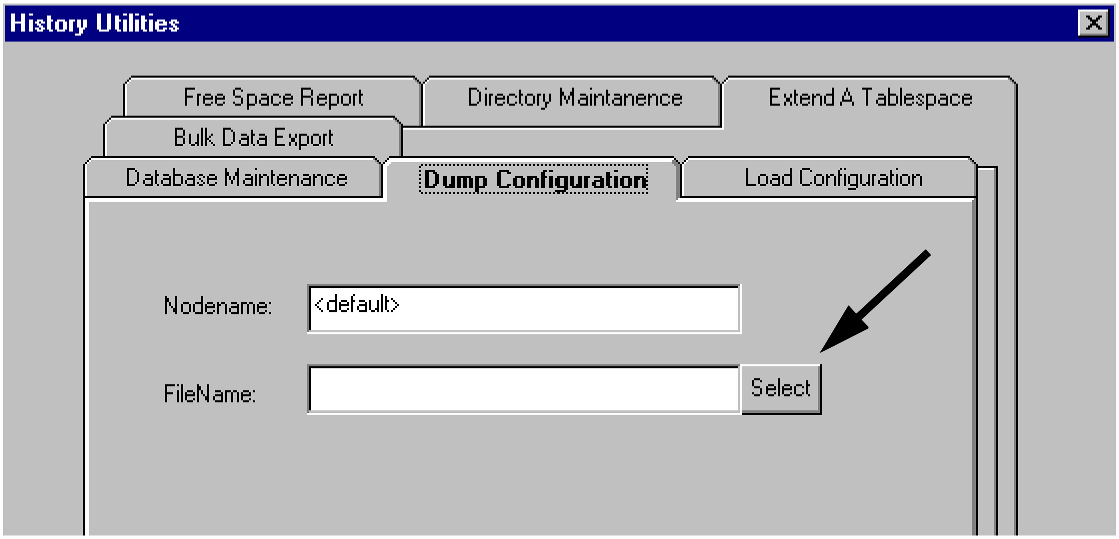
5.2.15 Dump & Load Configuration

The Dump Configuration and Load Configuration functions can be used when you build History logs via text files as described in Section 3.3, Building History Logs with Text Files. These functions provide a convenient graphical user interface for performing some of the basic `dbConfig` operations.

5.2.15.1 Dump Configuration

This function lets you write the History database configuration to a text file. You must first specify the location and file name for the dump file, and then perform the dump. The following example shows you how:

1. Click **Select** next to the FileName field, Figure 5-31.



Nodename: <default>

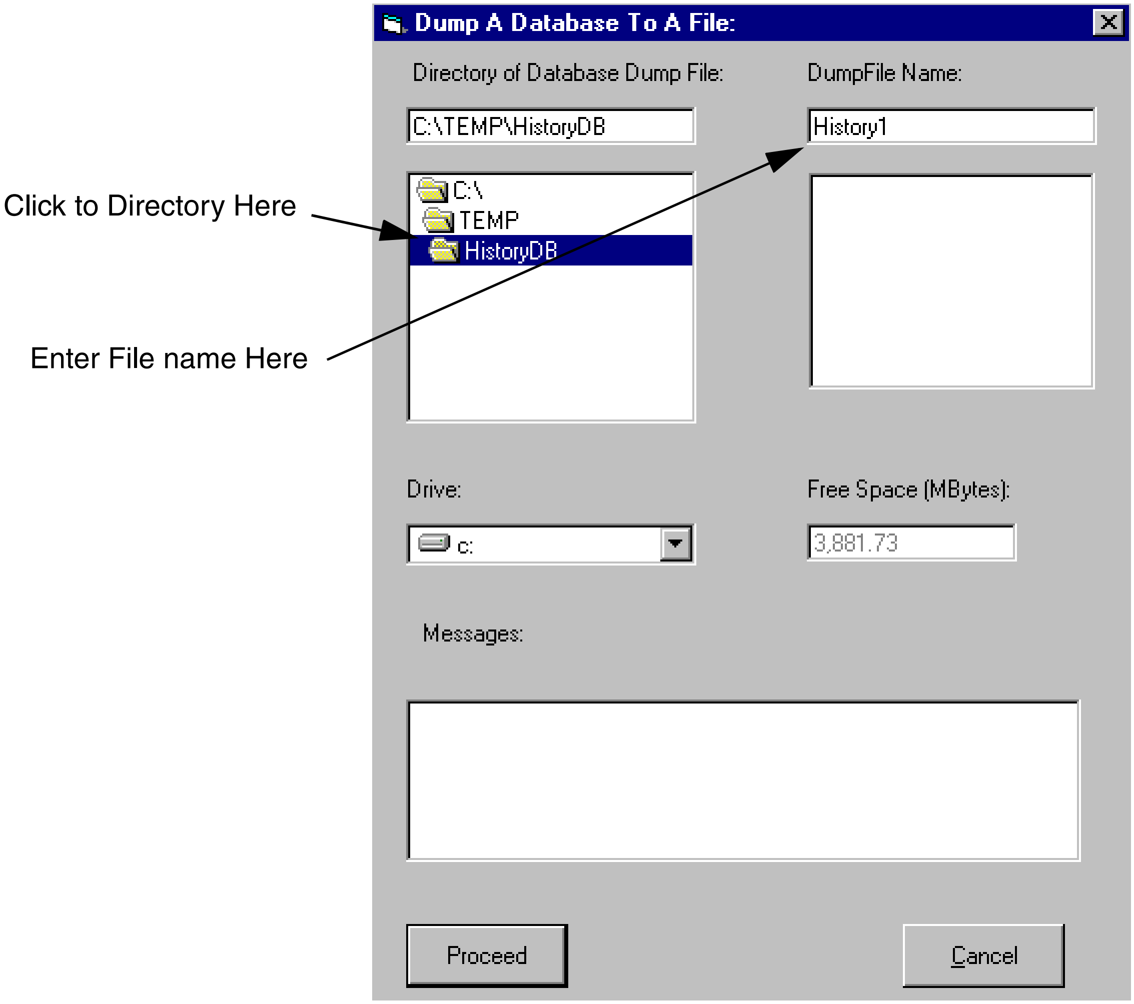
FileName:

Figure 5-31. Click Select to Start Path Specification for Dump File

2. This displays a dialog for specifying the path to the text file, Figure 5-32. Use the navigation window to specify the path to the text file, and then enter the text file name in the field labeled DumpFile Name. For instance, in Figure 5-32, a text file named History1 will be created in C:\TEMP\HistoryDB.

NOTE

The Dump utility will create the text file. The file does not have to exist ahead of time.



Directory of Database Dump File: C:\TEMP\HistoryDB

DumpFile Name: History1

C:\
TEMP
HistoryDB

Drive: c:

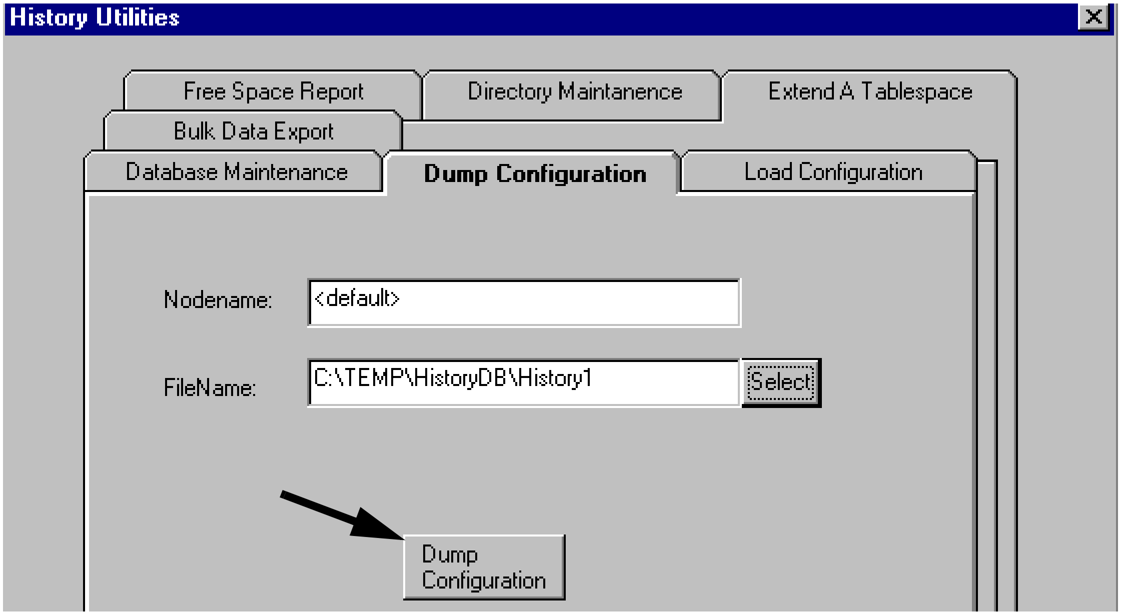
Free Space (MBytes): 3,881.73

Messages:

Figure 5-32. Specifying the Location and File Name

3. Click **Proceed** when you are finished.

4. To perform the dump, click **Dump Configuration**, Figure 5-33.

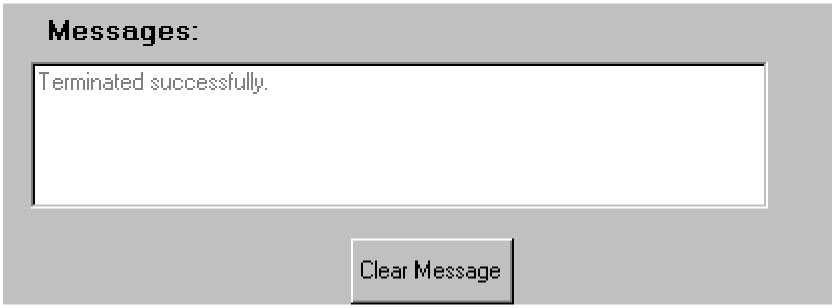


Nodename: <default>

FileName: C:\TEMP\HistoryDB\History1

Figure 5-33. Starting Database Dump

Check the Messages window to determine when the Dump is complete, Figure 5-34.



Terminated successfully.

Figure 5-34. Successful Database Dump Indicated in Message Window

For details on how to apply this utility for configuring History logs with text files, see Section 3.3, Building History Logs with Text Files.

5.2.15.2 Load Configuration

This function lets you load the History database from text files. Normally, this function is used to load data into a clean history database (that is, no other objects have been configured for that node), or to add more logs to an existing database. If objects that already exist in the database are present in the file being loaded, the Load Configuration function will report an error and continue creating other objects in the file. The existing object is not affected.

The load can be done with History running; however, it will take longer. Also, History must be re-started to enable the new logs to collect data. If this load fails because of disk space problems or Oracle tablespace problems, just repeat the procedure after you correct the problems.

Dropping the Current Database

As an option, you can use the `hsDBMaint` utility to drop the current database and create an empty database *before* you load the database from text files. Do this only when your current database contains bad or useless data as a result of testing or experimenting. Refer to Section 5.2.16, Create or Drop a Product Database for instructions on dropping and creating a database.

Before you use hsDBMaint to drop and create a database, you must first stop all Advant OCS software on the History node, and then restart the History node when you are finished.Refer to the *Advant Workplace Products Administrator's Guide* for details on how to do this.

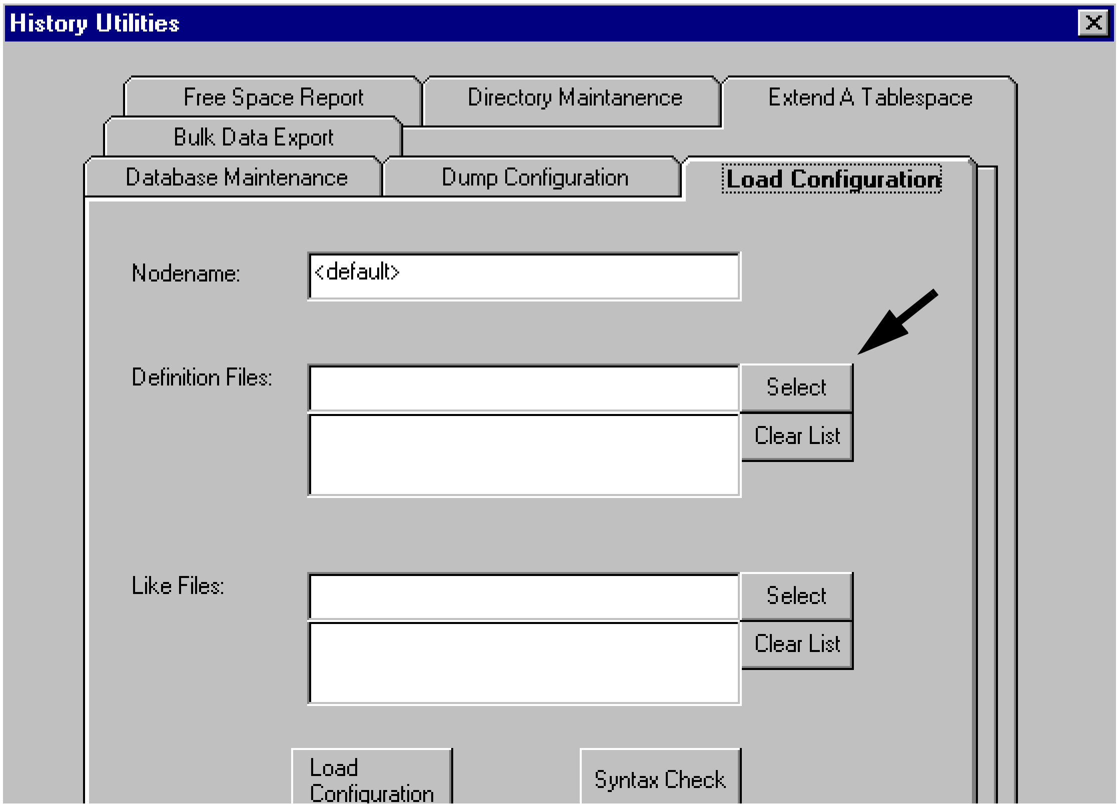
If necessary, use the Extend Tablespace function to extend tablespace, and Directory Maintenance function to increase the quota for file-based logs. The quota should slightly exceed the numbers provided by dbConfig.

Using the Load Configuration Function

You can check the syntax of any file before you load it. To do this, select the file and then click **Syntax Check**. If database errors are found (for instance the value of a high limit attribute is set higher than the high-high limit attribute), the file will not be able to be loaded unless the file is corrected, or the affected logs are deleted from the file.

To use the History Utility to load the database from the text file:

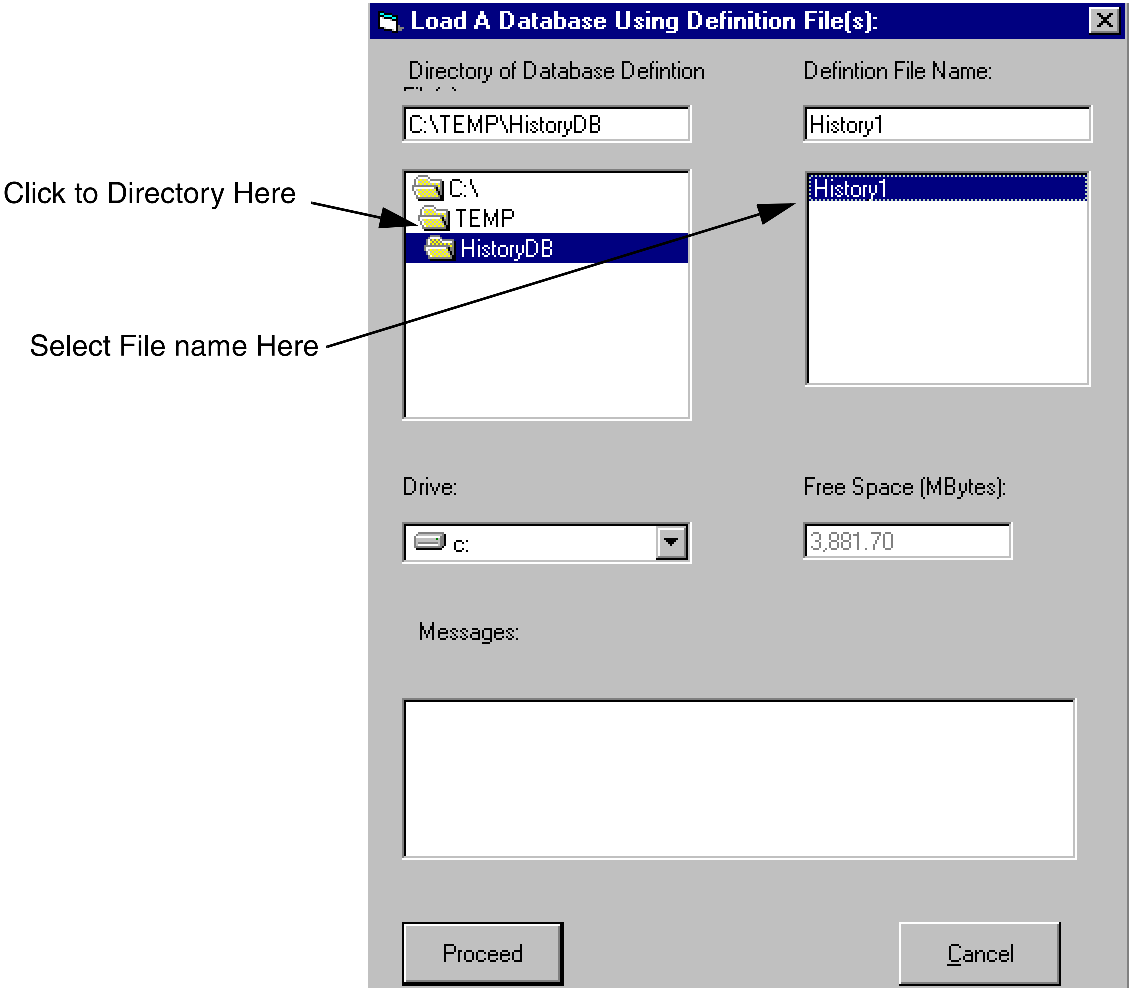
1. Click **Select** next to the Definition Files field, Figure 5-35.



Nodename: <default>

Figure 5-35. Click Select to Start Specification of Destination Files

This displays a dialog for specifying the text (Dump) file, Figure 5-36. Use the navigation window to navigate to the file, and then select the file from the list of files.



Click to Directory Here

Select File name Here

Directory of Database Defintion: C:\TEMP\HistoryDB

Defintion File Name: History1

Drive: c:

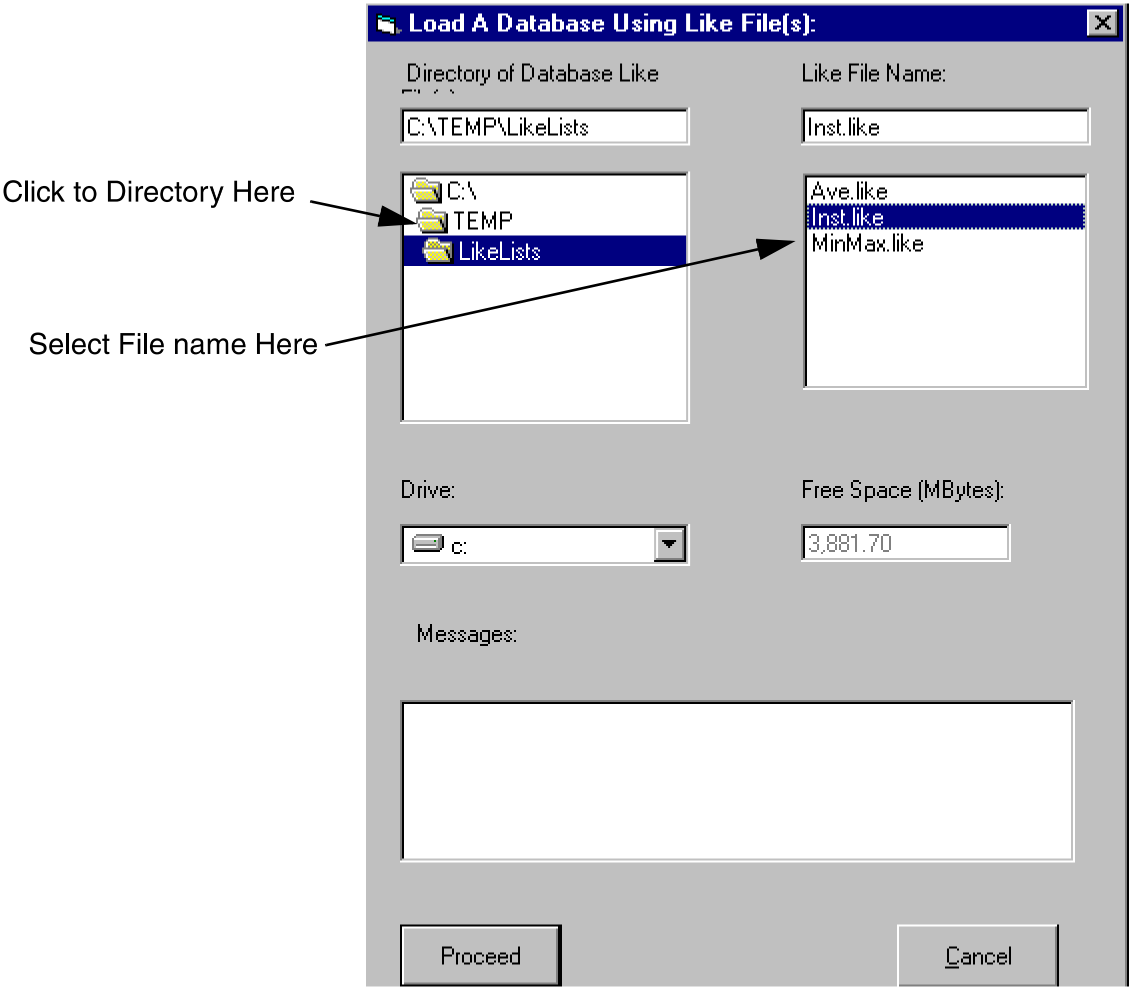
Free Space (MBytes): 3,881.70

Messages:

Figure 5-36. Specifying Location and name for Dump File

2. Click **Proceed** when you are finished.

3. Click **Select** next to the Like Files field. This displays a dialog for specifying the Like files, Figure 5-37. Use the navigation window to navigate to the files, select one file from the list, and then click **Proceed**. Repeat this for as many like files as required.



Load A Database Using Like File(s):

Directory of Database Like: C:\TEMP\LikeLists

Like File Name: Inst.like

C:\
TEMP
LikeLists

Ave.like
Inst.like
MinMax.like

Drive: c:

Free Space (MBytes): 3,881.70

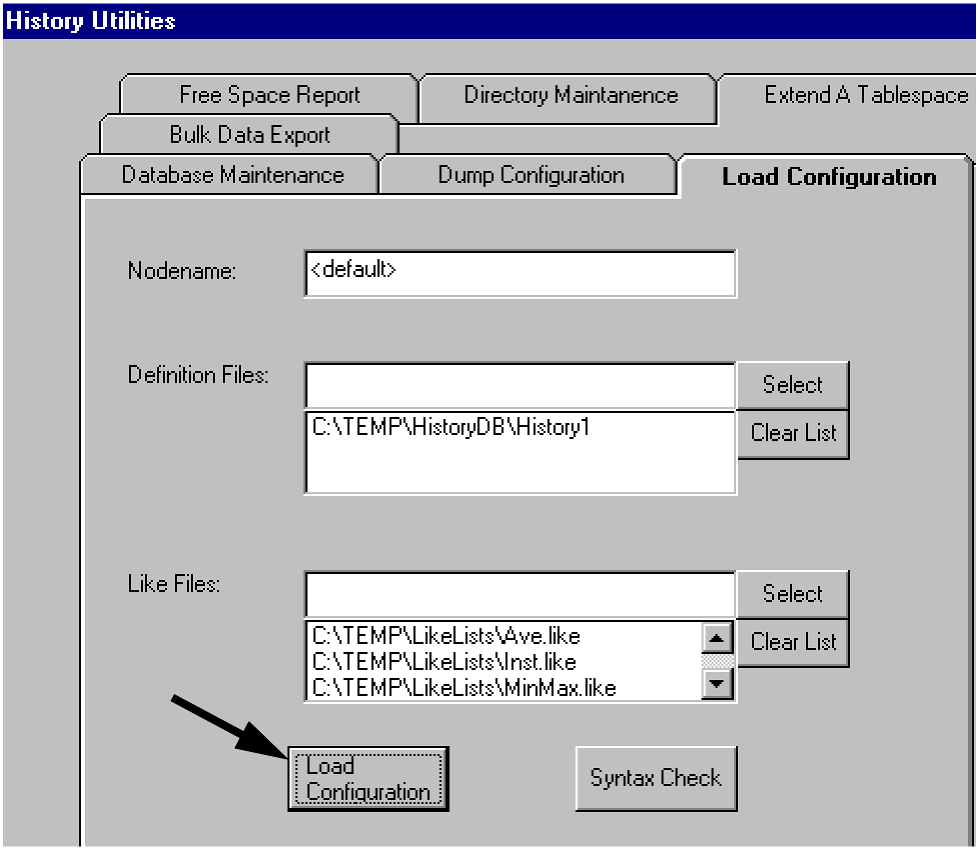
Messages:

Figure 5-37. Specifying the Like Files

When you are finished selecting the Definition and Like files, the Load Configuration utility will list your selections, Figure 5-38.

If you want to check the syntax for the selected files, you can do that now by clicking **Syntax Check**. You are not required to check syntax now if you run the output report as described in Section 3.3.1.5, An Example Application of dbConfig.

4. To perform the load, click **Load Configuration**.



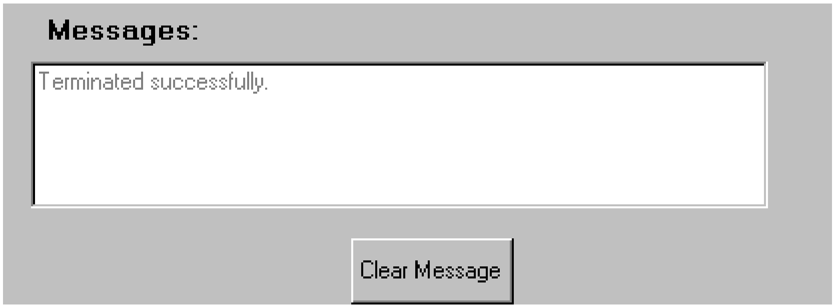
Nodename: <default>

Definition Files:
C:\TEMP\HistoryDB\History1

Like Files:
C:\TEMP\LikeLists\Ave.like
C:\TEMP\LikeLists\Inst.like
C:\TEMP\LikeLists\MinMax.like

Figure 5-38. File Selection Completed

5. Check the Messages window to determine when the load is complete, Figure 5-39.



Terminated successfully.

Figure 5-39. Successful Load Configuration Indicated in Message Window

NOTE

If the load fails due to lack of quota for flat files, after configuring more space you must repeat the dump procedure as described in Section 5.2.15.1, Dump Configuration, or manually update the dump file with a text editor to reflect the new sizes.

5.2.16 Create or Drop a Product Database

Use this function to create or drop (delete) a database. This is an off-line maintenance function.

When you drop a database, all configuration data and stored log data is deleted. As an example, you may use this function while you are configuring your History database, and you determine that it would be better to delete your work up to this point and start from scratch. In this case you would drop the current database, and then create a new one.

NOTE

When you drop a database, all configuration data and log data is lost. If you need to drop a database after History has been running (database has live data), be sure to archive any data that you do not want to lose.

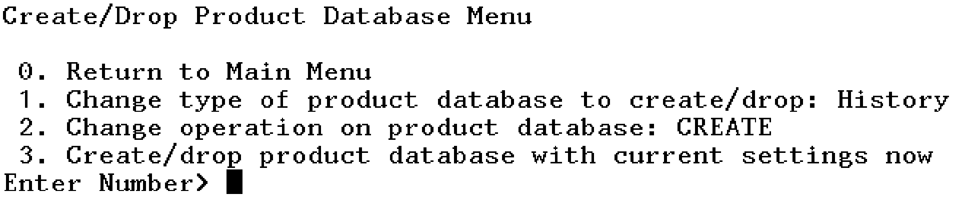
To create or drop a product database:

1. Invoke the `hsDBMaint` command from the DOS command prompt:

   ```
   c:> “%HS_HOME%” \bin\hsDbMaint.exe
   ```

   This displays the off-line database maintenance menu, Figure 5-2.

2. From this menu.choose **Create/Drop Product Database**. This displays the Create/Drop Product Database menu, Figure 5-40.



```
Create/Drop Product Database Menu

 0. Return to Main Menu
 1. Change type of product database to create/drop: History
 2. Change operation on product database: CREATE
 3. Create/drop product database with current settings now
Enter Number> █
```

Figure 5-40. Create/Drop Product Database Menu

This menu shows you the current settings for type of database to create/drop, and the operation to perform (create or drop). You can change the current settings before you execute the create or drop operation, or you can return to the main menu without performing any of these functions.

If you need to change any of the settings, choose the corresponding option from this menu and follow the dialog. For type of database you can choose:

- History Database

For operation to perform you can choose:

- Create
- Drop

When you finish with the dialog, you will return to the Create/Drop Product Database menu and the current settings will be changed according to the dialog. When the settings are correct, choose the Create/drop product database with current settings now option. When the operation is finished, the Main Menu will be displayed again.

If you need to return to the database maintenance main menu without invoking any of these functions, choose the **Return to Main Menu** option (0).

NOTE

You can use the History Utilities to drop a database. This function ia available under the **Database Maintenance** tab, Figure 5-41. Select the type of database to drop (**History** or **PDL**), then click **Drop History**.

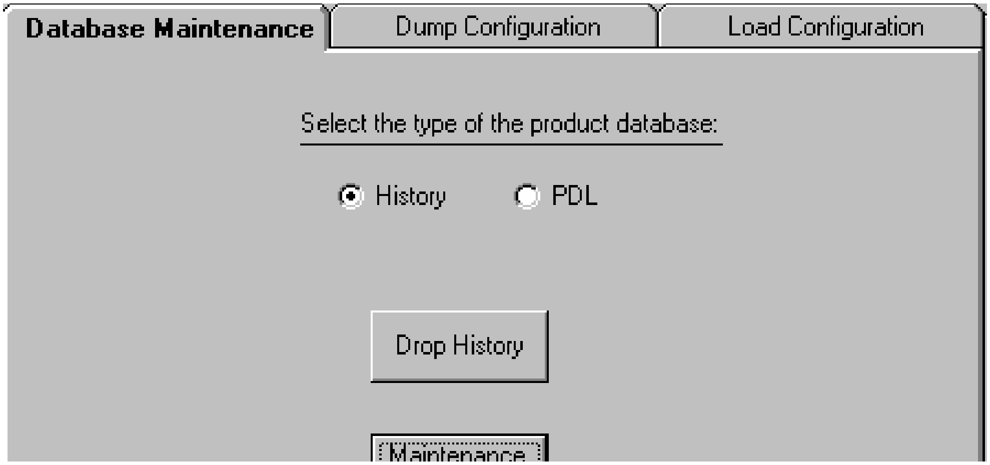


Figure 5-41. Database Maintenance Tab

5.2.17 Clean History Database

Use this function to clean the History database. This is an off-line maintenance function.

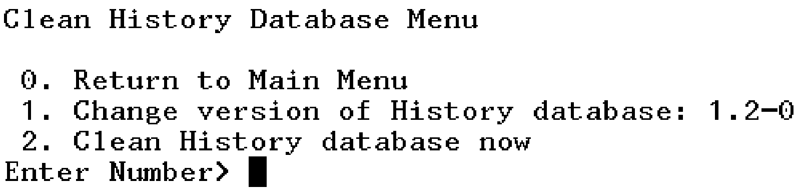
To clean the History database:

1. Invoke the `hsDBMaint` command from the DOS command prompt:

   ```
   c:> “%HS_HOME%” \bin\hsDbMaint.exe
   ```

   This displays the off-line version of the database maintenance menu, Figure 5-2.

2. From this menu.choose **Clean History Database**. This displays the Clean History Database Menu, Figure 5-42.



```
Clean History Database Menu

 0. Return to Main Menu
 1. Change version of History database: 1.2-0
 2. Clean History database now
Enter Number> █
```

Figure 5-42. Clean History Database Menu

3. To clean the History database, enter **1** and press the ENTER key.

   This cleans up the database, and then returns you to the Clean History Database menu.

4. To return to the Main Menu, enter **0** and then press the ENTER key.

5.2.18 Restore or initialize History Logs

Use this function to:

- redefine missing tables for Oracle logs (storage type (Oracle) and missing files for file-based logs storage type (TYPE1 or TYPE2).
- initialize all log tables and log file essentials, and erasing all runtime data.

This is an off-line maintenance function. To restore or initialize History logs:

1. Invoke the `hsDBMaint` command from the DOS command prompt:

   ```
   c:> "%HS_HOME%" \bin\hsDbMaint.exe
   ```

   This displays the off-line version of the database maintenance menu, Figure 5-2.
2. From this menu, choose **Restore/Initialize History Logs**.

5.2.19 Reindex Logs

Use this function to reindex logs.This is an off-line maintenance function.To restore or initialize History logs:

1. Invoke the `hsDBMaint` command from the DOS command prompt:

   ```
   c:> "%HS_HOME%" \bin\hsDbMaint.exe
   ```

   This displays the off-line database maintenance menu, Figure 5-2.
2. From this menu, choose **Reindex Logs**.

5.2.20 Fill Synchronous Entry Tables

Use this function to fill synchronous entry tables.This is an off-line maintenance function.To fill synchronous entry tables:

1. Invoke the `hsDBMaint` command from the DOS command prompt:

   ```
   c:> "%HS_HOME%" \bin\hsDbMaint.exe
   ```

   This displays the off-line database maintenance menu, Figure 5-2.
2. From this menu.choose **Fill Synchronous Entry Tables**.

5.2.21 Purge History Data

If, for some reason, your system time is set ahead this will cause History to collect data "in the future". For instance, this would occur if the time was set wrong. If this occurs, even if you fix the time, History will not collect any more data because it will not collect data that is older than the most current data collected. In this case you must use the Purge History Data function to delete all data that have time stamps in the future.

This is an off-line maintenance function. You can either use the -purge command line option, or use hsDBMaint.

To use hsDBMaint:

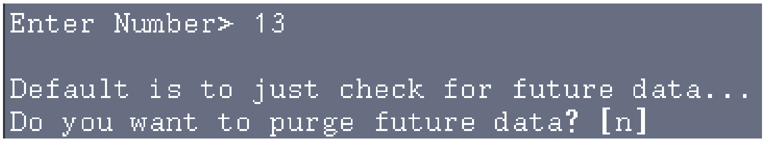
1. Invoke the `hsDBMaint` command from the DOS prompt:

   ```
   c:> "%HS_HOME%" \bin\hsDbMaint.exe
   ```

   This displays the off-line version of the database maintenance menu, Figure 5-2.

2. From this menu, choose **Purge History Data**.

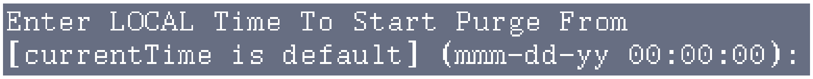
When you invoke this function you are prompted as to whether you want to actually purge the data, or just check which logs have future data, Figure 5-43. The default selection is to just check, and not purge. After you specify whether or not you want to purge, you are prompted for a time to compare data time stamps with, Figure 5-44. The default is the current time. When the function is finished, a table is displayed that indicates the number of logs checked, and the number of logs found to have future data, Figure 5-45. When History detects a time change of greater than 75 minutes (this is configurable via the environment variable HS_MAX_TIME_CHANGE), History will log an operator message stating that fact.



```
Enter Number> 13

Default is to just check for future data...
Do you want to purge future data? [n]
```

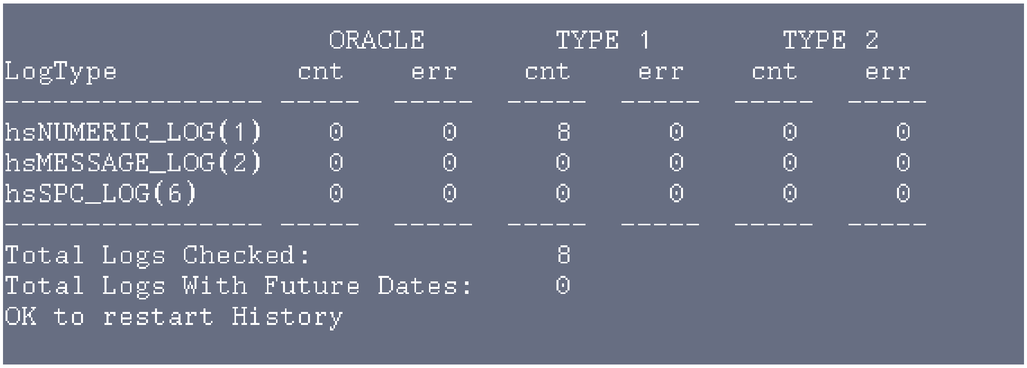
Figure 5-43. Prompt to Purge or Check

.



```
Enter LOCAL Time To Start Purge From
[currentTime is default] (mmm-dd-yy 00:00:00):
```

Figure 5-44. Prompt for Time

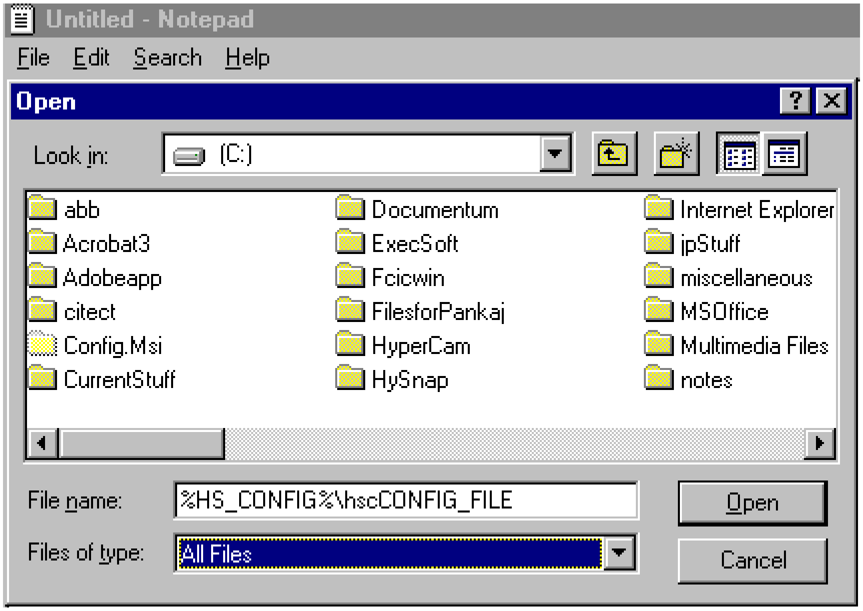


```
                    ORACLE        TYPE 1        TYPE 2
LogType           cnt    err    cnt    err    cnt    err
---------------- -----  -----  -----  -----  -----  -----
hsNUMERIC_LOG(1)    0      0      8      0      0      0
hsMESSAGE_LOG(2)    0      0      0      0      0      0
hsSPC_LOG(6)        0      0      0      0      0      0
---------------- -----  -----  -----  -----  -----  -----
Total Logs Checked:               8
Total Logs With Future Dates:     0
OK to restart History
```

Figure 5-45. Purge Future Data Results

5.3 Configuring OPC Server Access

The OPC server that History is connected to is specified in the text file hscCONFIG_FILE. You can open this file with a text editor such as Notepad. Access to this file is via the `%HS_CONFIG%` environment variable which points to the directory where the file resides. To open the file, in Notepad or similar text editor, specify the file path in the Open dialog, Figure 5-46.



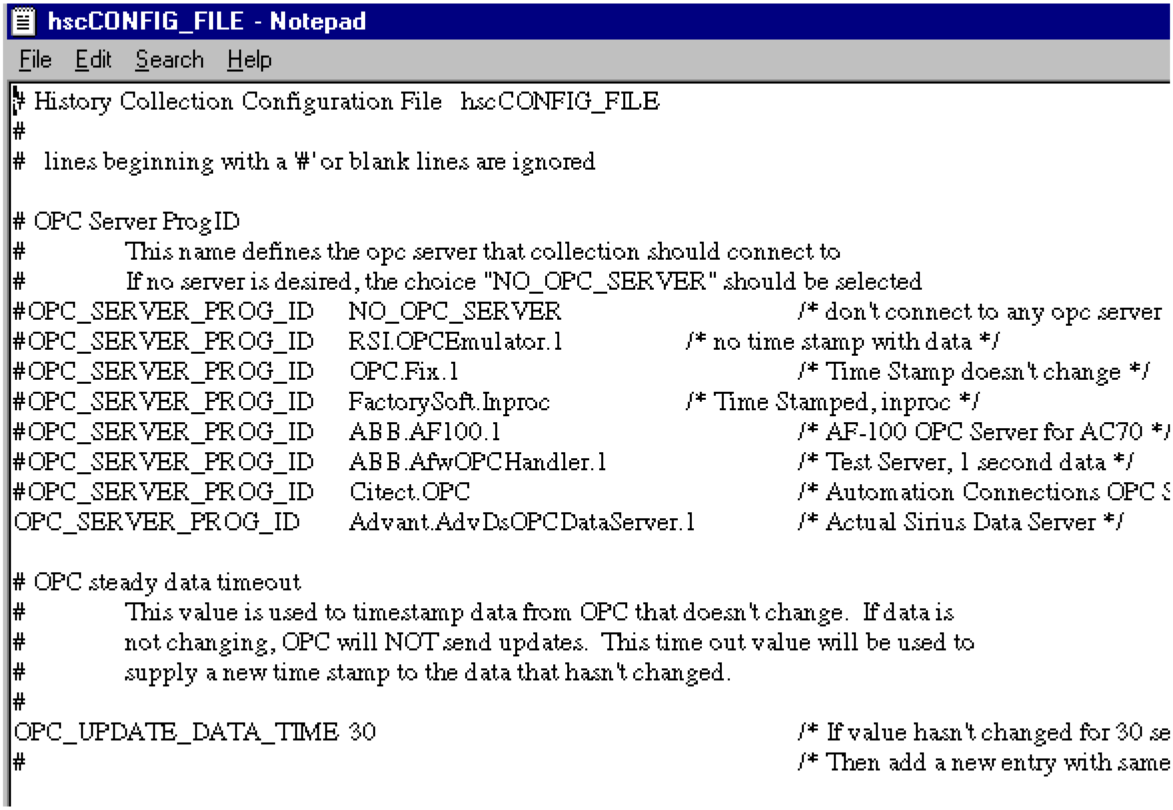
Look in: (C:)

File name: %HS_CONFIG%\hscCONFIG_FILE

Files of type: All Files

Figure 5-46. Specifying the Path to hscCONFIG_FILE

The contents of the file are shown in Figure 5-47.



```
# History Collection Configuration File   hscCONFIG_FILE
#
#   lines beginning with a '#' or blank lines are ignored

# OPC Server ProgID
#            This name defines the opc server that collection should connect to
#            If no server is desired, the choice "NO_OPC_SERVER" should be selected
#OPC_SERVER_PROG_ID    NO_OPC_SERVER                         /* don't connect to any opc server
#OPC_SERVER_PROG_ID    RSI.OPCEmulator.1              /* no time stamp with data */
#OPC_SERVER_PROG_ID    OPC.Fix.1                             /* Time Stamp doesn't change */
#OPC_SERVER_PROG_ID    FactorySoft.Inproc             /* Time Stamped, inproc */
#OPC_SERVER_PROG_ID    ABB.AF100.1                           /* AF-100 OPC Server for AC70 */
#OPC_SERVER_PROG_ID    ABB.AfwOPCHandler.1                   /* Test Server, 1 second data */
#OPC_SERVER_PROG_ID    Citect.OPC                            /* Automation Connections OPC S
OPC_SERVER_PROG_ID     Advant.AdvDsOPCDataServer.1           /* Actual Sirius Data Server */

# OPC steady data timeout
#            This value is used to timestamp data from OPC that doesn't change.  If data is
#            not changing, OPC will NOT send updates.  This time out value will be used to
#            supply a new time stamp to the data that hasn't changed.
#
OPC_UPDATE_DATA_TIME 30                                      /* If value hasn't changed for 30 se
#                                                            /* Then add a new entry with same
```

Figure 5-47. hscCONFIG_FILE

The server that History is connected to is the one that is not *commented* out (does not have # character at the beginning of the line). As shown in Figure 5-47, the default server is `Advant.AdvDsOPCDataServer.1`.

Index

A

adding secondary logs 45
Advant
  Reports 6
Advant Trend software 107
alarm 71
Alarm Limit
  Low Low 72
alarm limit
  high 72
  high high 72
  low 72
alignment 62
API 58
aspect 10, 29
attributes
  alarm 71
  calculation 59
  deadband 66
available composite logs 18

B

bad data quality limit 63
bulk data export 143

C

calculation algorithm 64
Calculations 76
calculations 64, 78
Cascade Attributes 135
clean history database 165
collection mode 58
  API 58
  OPC 58
Collection Type 58
collection type 58
color key 21
Compaction Ratio 69
compaction ratio 69
COMPACTION_RATIO 122
composite log 17
  adding a new composite log 32
  adding primary and secondary logs 35
  deleting logs from 54
  description 35
composite log name 34
configuration status 23
copying a composite log 45

D

data
  collection 4
  entry 83
  presentation 6
  purge 167
  retrieval 5
data collection 59
  examples 72
  starting/stopping 103
  storing calculated values 76
  storing calculated values with compaction 79
  storing instantaneous values using compaction 74
  storing instantaneous values, no compaction 73
data source 35
Database
  create 163
database
  drop&create 163
dbConfig 87, 90, 99
deadband 66
deadband compaction 66, 70
deadband storage interval 70
disk space requirements 85
display
  trend 6

dump 90

E
effective log period 68
engineering units 71
  maximum value 71
  minimum value 71
entry status 83
entry tables report 119
error indication 23
error message 23
EST_LOG_TIME_PER 69, 122
Estimated Period 70
estimated period 70
expected ratio 70

F
file storage 82
file-based storage 126
free space report 116

G
getting started
  configuration 15
  history data access 105

H
history 1
history builder window 18

I
Interpolation 110
interpolation 110

L
like statement 91
Log
  Capacity 60
  Period 60
log
  activation/deactivation 103
  capacity 65
  directories 126
  name 105
  period 60
  starting and stopping a log 56
  status 57
log attribute
  attribute value dialog 55
  tabs 23
log attributes
  how to configure 54
  numeric 57
Log Entry
  Time Stamp 83
  Value 83
log entry
  status 83
  time stamp
    local time 83
    universal time (UTC) 83
log name 105
log period 60

N
Name 57
numeric log
  example
    calculations 51
    secondary for varying time span 44
    simple 35
numeric log name 57

O
object 10
object status 85

P
presentation

    maximum value 71
    minimum value 71
primary log 35
purge 167

R

Reset Object Status 133

S

sample blocking rate 65
save composite log as 45
save ss 45
Seamless retrieval 108
seamless retrieval 108
secondary log 44
SQL 3
stagger collection 136
start state 58
start time 62
starting a log 56
state set 71
status 23, 57
stopping a log 56
storage interval 63
Storage type 82
    TYPE1 82
    TYPE2 82
storage type 60, 82

T

Tablespace
    Inform_HS_Runtime 123
Text File 87
tree view 20
    selecting log attributes to display 31
trend display 107
    Advant Trend 6
trend displays 6

U

update deadband ratio 121
user interface 18